高等院校计算机教育系列教材



ASP.NET Web 开发教程

程不功 编著

- 内容简明扼要，突出知识要点。
- 知识点新，突出实践教学，强化能力培养。
- 深入浅出地阐述设计思想和相关理论。
- 本书内容涵盖以ASP.NET 3.5 Web开发的各项关键技术(C#语言)。

清华大学出版社

ASP.NET Web 开发教程

程不功　编著

清华大学出版社
北　京

内 容 简 介

这是一本 ASP.NET 3.5 Web 开发的教程(C#版)。全书共分 7 部分，包括 29 章和两个附录。内容涵盖 ASP.NET 基础、浏览器端开发、服务器端开发、母版页与角色管理、ASP.NET Ajax 技术、XML Web 服务以及综合示例等。

本书的特点是，以 Web 应用开发为主线组织内容；通过大量实例深入浅出地讲解技术，引导读者利用 ASP.NET 3.5 快速开发出功能强大、运行可靠而且易于扩展的系统。

本书的主要对象是计算机应用、信息专业的大专院校学生，初、中级程序员或者 Web 的业余爱好者。对于一些想要自行开发 Web 的自学者，本书也是一本很好的自学教材。

图书在版编目(CIP)数据

ASP.NET Web 开发教程/程不功编著. --北京：清华大学出版社，2011.9

ISBN 978-7-302-26722-5

Ⅰ. ①A…　Ⅱ. ①程…　Ⅲ. ①网页制作工具—程序设计—教材　Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字(2011)第 177025 号

责任编辑：邹　杰
装帧设计：杨玉兰
责任校对：李玉萍
责任印制：

出版发行：清华大学出版社　　**地　址：**北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：**100084
社 总 机：010-62770175　　**邮　购：**010-62786544
投稿与读者服务：010-62776969, c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015, zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185×260　**印　张：**29.5　**字　数：**714 千字
版　　次：2011 年 9 月第 1 版　**印　次：**2011 年 9 月第 1 次印刷
印　　数：1~4000
定　　价：49.00 元

产品编号：

前　　言

微软 2008 年推出的 ASP.NET 3.5 是在 ASP.NET 2.0 的基础上发展起来的，一方面它进一步完善了 2.0 版本，另一方面又增添了很多最新技术，但仍然与 2.0 版本兼容，甚至允许两种版本安装在同一台机器上。而且微软公司已经公开承诺允许长期免费使用这两个系统。

本书的特点是，以 Web 应用开发为主线组织内容；通过大量实例深入浅出地讲解技术，引导读者利用 ASP.NET 3.5 快速开发出功能强大、运行可靠而且易于扩展的系统。

本书不是一本“傻瓜书”，也不是一本“应用手册”，而是一本“教程”。就是说，书中不仅要说明技术“是什么”，还要说明技术产生的历史背景、解决的思路和方法，以及相关理论。

学习本书之前最好已经学过一门程序设计语言(C、C++、Java、VB 等都可以)，如果学过 C#.NET 当然更好，没有学过 C#.NET 也没有多大关系，必要时可参阅本书的附录。

本书共分七大部分，包括 29 章和两个附录。

第一部分：ASP.NET 基础。介绍.NET 框架的组成、作用以及 ASP.NET 网站的逻辑结构。

第二部分：浏览器端开发。浏览器端的开发技术近几年来有很大发展，网页质量已经成为网站竞争的重要因素。这一部分讲授 HTML、CSS、动态 HTML(即 DHTML)以及利用 jQuery 设计动画等四方面的内容。

第三部分：服务器端开发。这是本书的重点。ASP.NET Web 应用程序(网站)就是一种基于服务器的系统，它包括的内容很多，因此用 12 章的篇幅进行讲解，主要讲授 ASPX 网页以及各种服务器控件的使用方法；各种数据访问技术，包括 ADO.NET 的系统结构，对数据库的连接、显示、编辑与同步，以及存储过程、数据缓存的方法；最后介绍 LINQ 这项最新技术及其应用等。

第四部分：母版页与角色管理。网站安全、网站显示风格等都是网站开发中必须注意的主题。在这部分中将站在全局的高度来讲述网站的显示风格、网站导航、网站安全以及网站的个性化服务等四方面的内容。

第五部分：ASP.NET Ajax 技术。用 3 章篇幅主要讲授 Ajax 原理、ASP.NET Ajax 技术以及 Ajax ToolKit 的使用等三方面的内容，使读者在学会原理的基础上，能够对传统的 Web 网站进行 Ajax 功能升级。

第六部分：Web 服务。主要通过几个典型实例让读者掌握 XML Web 服务的调用方法，以及创建 XML Web 服务网站两方面的问题。

第七部分：综合示例。在这里选择了网上招聘及留言板、快速开发动态数据驱动网站、创建电子商务网站三个典型项目，通过讲述它们的设计方法，使读者进一步掌握利用前面所学内容综合开发系统的实际本领。

本书是《ASP.NET 2.0 动态网站开发教程》的姊妹篇。2006 年上半年本书的作者曾经出版了《ASP.NET 2.0 动态网站开发教程》，2008 年又出版了它的第 2 版，这两本书受到

了不少读者的喜爱(4 年中已经连续印刷了 14 次)，同时也要感谢很多读者提出的批评和建议，现在这本书保留了前两本书的撰写风格和部分内容，因此可以说本书既站在 Web 发展的前沿又保持了原书的优点。

本书的主要读者对象是计算机应用、信息专业的大专院校学生，初、中级程序员或者 Web 的业余爱好者。

编　者

目　　录

第一部分　ASP.NET 基础

第二部分　浏览器端开发

第三部分 服务器端开发

第四部分 母版页与角色管理

第五部分　ASP.NET Ajax

第六部分　Web　服　务

第七部分 综 合 示 例

第一部分

ASP.NET 基础

为了学会并掌握 ASP.NET 系统，需要对系统的发展过程、系统的整体概貌以及.NET 框架的基础结构有一个初步的了解。这一部分讲述以下几方面的问题。

- Web 的发展简史。
- .NET 框架与 ASP.NET 发展的简要过程。
- .NET 框架的基础结构。
- ASP.NET 网站的逻辑结构与网站组成。
- 创建 ASP.NET 网站的方法。

第 1 章　.NET 框架简介

本章讲授的内容包括三部分。

- Web 以及 ASP.NET 发展历史的简要回顾。
- .NET 框架的基础结构。
- XML 语言。

后面两个问题是重点，在后续的讲授中将经常联系到这两方面的内容。

1.1　Web 发展历史的简要回顾

Web 的概念是 Tim Berners-Lee 于 1980 年在瑞士提出来的。1991 年 8 月出现了第一台 Web 服务器，到今天才不过 20 年的历史，Web 出现后的发展速度、影响的范围都远远超出了人们的预想。目前，它已经深入到人类社会，并广泛应用于人们的生活之中。

Web 通常由浏览器、服务器(通常还包括数据库)组成。其一部分是浏览器或其他数字设备，另一部分是服务器。最初的 Web 只能由服务器发送简单的单页静态纯文本，现在已经发展到支持音频、视频这样丰富内容的信息，还常常执行一些复杂的基于因特网的应用程序。在这些应用程序中，服务器与浏览器之间频繁地进行通信、交互以处理大量的数据，实际上已经成为一种分布式的应用系统。一个 Web 应用程序既是 Web 应用的基本单位，也是程序部署的基本单位，有时又称为网站。例如电子商务、电子政务、远程教育、网上银行、网上招聘、信息咨询等一些大大小小的网站都属于 Web 应用的范围。

1.1.1　从静态网页发展到动态网页

早期的 Web 服务器发送出的是静态网页，虽然网页中包括有文字和图片，但是只要不改变设计，网页的内容是不会变化的。对静态网页的访问过程如下。

(1) 客户通过浏览器向服务器申请 URL 页面。

(2) 服务器向客户送出被申请的页面。

(3) 浏览器下载并解释传来的页面并显示在浏览器中。

(4) 断开客户与服务器之间的联系。

整个过程比较简单，到浏览器下载完页面时为止，整个过程就结束了。用于发布静态网页的网站设计比较简单。这种设计对于早期的网站来说也许已足够。因为早期使用网站的大多是一些科学工作者，他们关注的重点只是交流有关科学技术的内容。

随着因特网应用领域的扩展，各种不同类型的客户加入到网络中来，不少客户很快就提出了新的要求。例如，有的客户提出，能不能代我查阅一下我银行存款的变化情况？要满足类似这样的需求，服务器的工作就不那么简单了。它首先要查阅银行账户，进行必要的计算和统计，再将结果反馈给客户。这就是说，服务器在回答问题前必须先执行一些相关的程序。这段程序不仅应能回答客户的问题，还要能够保障客户的信息安全，防止其他人进行查阅或破坏。

类似这种网页的输出内容将随程序执行的结果而有所不同，这样的网页被称为“动态网页”。访问动态网页的过程如下。

(1) 客户通过浏览器向服务器申请 URL 页面。

(2) 服务器接收请求，并处理网页上的代码。

(3) 将代码的处理结果转换成 HTML 代码后向客户送出。

(4) 浏览器下载并解释传来的页面并显示在浏览器中。

(5) 服务器断开与客户的联系并转向其他客户，以便提供新的服务。

和静态网页相比，动态网页的处理上多了一个处理代码的过程。用什么方式来处理代码，在不同的历史时期采用了不同的技术，大体上可以划分为 3 个阶段。

1.1.2 动态网页发展的几个阶段

1. CGI 阶段

CGI 是英文 Common Gateway Interface 的缩写，代表服务器端的一种通用(标准)接口。每当服务器接到客户更新数据的要求以后，利用这个接口去启动外部应用程序来完成各类计算、处理或访问数据库的工作，处理完后将结果返回 Web 服务器，再返回浏览器。外部应用程序是用 C、C++、Perl、Pascal、Java 或其他语言编写的，程序运行在独立的地址空间中。具体情况如图 1.1 所示。

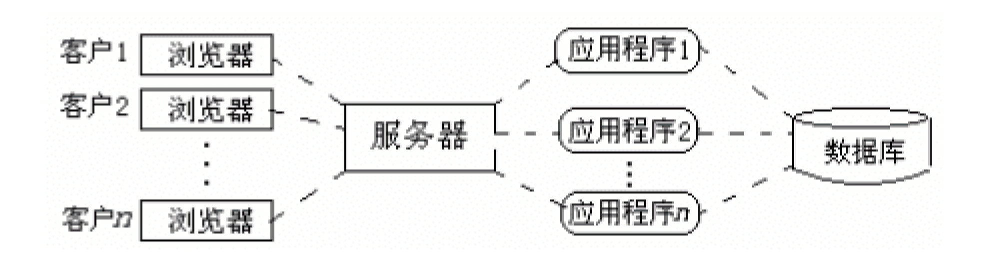


图 1.1 CGI 示意图

后来出现了 ISAPI(用于 Internet Explorer 浏览器)或 NSAPI (用于 Netscape 浏览器)技术，其功能与 CGI 相同，但技术方面有些改进。外部应用程序改用动态链接库(DLL)，被载入 Web 服务器的地址空间运行，并且用“线程”代替“进程”，因而显著地提高了运行效率。但不论是 CGI 还是 ISAPI 或 NSAPI，都需要编写外部应用程序，而编写外部应用程序并不是一件容易的事情。从开发人员的角度来讲，这种开发方式并没有带来开发上的方便。

2. 脚本语言阶段

在脚本语言阶段出现了许多杰出的脚本语言，如 ASP、PHP、JSP 等。脚本语言的出现大大简化了动态网站开发的难度，特别是 ASP 和 PHP 使用简单、功能强大，成为许多网站开发者的首选。

JSP 与 ASP 的程序结构非常相似。它的主要特点是在传统的 HTML 网页文件中加入 Java 程序段和使用各种各样的 JSP 标志(Tag)，构成 JSP 网页。Web 服务器在接收客户的访问要求时，首先执行其中的程序片段，并将执行结果以 HTML 格式返回客户。

3. 组件技术阶段

ASP.NET 和 Java(J2EE)技术是组件技术阶段的代表。这是一个由类和对象(组件)组成的完全面向对象的系统，采用编译方法和事件驱动方式运行。系统具有高效、高可靠、高

可扩展性的特点。详细情况将在下面各章节中重点讲述。

1.2 ASP.NET 发展的简要历史

进入 21 世纪前夕，微软公司鲜明地提出了.NET 的发展战略，确定了创建下一代 Internet 平台的目标。什么是下一代 Internet 平台？下一代 Internet 平台的主要特征之一就是，它将无处不在，世界上任何一台智能数字设备都有可能通过宽带连接到因特网上。因此，作为下一代 Internet 平台应该实现以下要求。

- 为各种类型的客户服务。不仅能为现有的计算机、手提式计算机、移动电话等客户服务，还要能为未来可能加入因特网的智能设备(如电视机、电冰箱、洗衣机等)提供服务。
- 强大的交互和运算能力。
- 丰富的表现能力。
- 跨平台交换数据的能力。
- 快速设计和部署的能力。
- 强有力的安全保障能力。

在这种思想的指导之下，微软公司首先设计出了.NET 框架(Framework)，然后在框架的基础上创建了多种应用程序的开发平台。

1.3 .NET 框架(Framework)的发展过程

1.3.1 什么是.NET 框架

什么是框架？框架是从建筑行业引过来的名字。在建筑行业中，它代表建筑物的骨架或基础架构。对于计算机软件来说，框架是一簇技术的集合，是应用程序开发的基础，它为应用程序提供了大量的服务。如提供了运行环境和基层管理；为应用程序的设计提供了大量可重用的类以及代码等。

Microsoft .NET Framework 框架是一个综合性的开发平台，在这个平台之上既可以开发 Windows 桌面应用系统、也可以开发 ASP.NET Web 应用系统、Web Service 服务系统以及 Mobile 移动式(无线)网络应用系统。

当各类应用程序以.NET Framework 框架为基础上进行开发时，能够借助于框架提供的服务从高起点出发，高速度地前进，不仅可以大大简化开发过程，还可以缩短开发周期。很多单靠应用设计本身难以解决的难题，在.NET Framework 服务的支持下，都能够迎刃而解。

1.3.2 .NET 框架的发展过程

.NET Framework 的发展可以分为两个阶段。

第一阶段：2000—2003 年微软先后推出了.NET Framework 1.0 和.NET Framework 1.1

版本，并在两种版本之上建立了 ASP.NET 1.0 和 ASP.NET 1.1，这两个版本都是独立的系统。

第二阶段：2005—2008 年微软先后推出了.NET Framework 2.0、.NET Framework 3.0、.NET Framework 3.5、.NET Framework 3.5 SP1 几种版本，在这些版本之上分别创建了 ASP.NET 2.0、ASP.NET 3.5 等。.NET Framework 的各种版本的关系如图 1.2 所示。

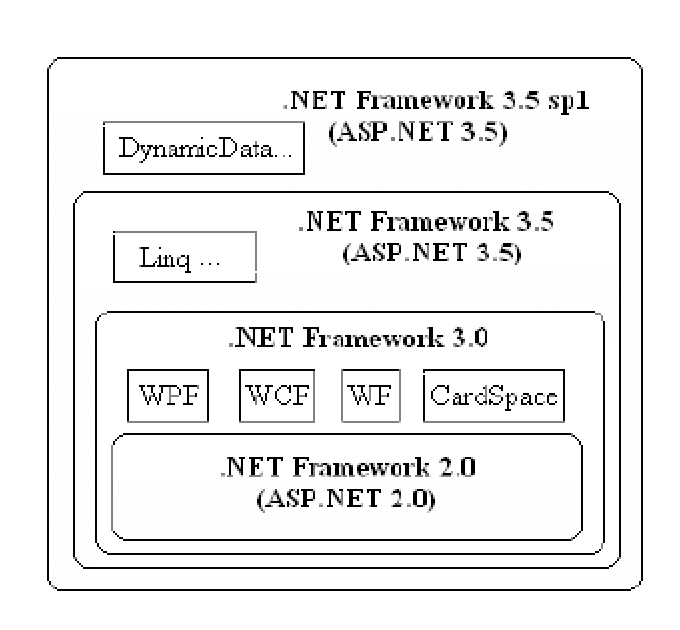


图 1.2 .NET Framework 从 2.0 到 3.5 版本的发展过程

图中的小方框代表新版本中增添的技术。新版本在原有版本的基础上发展，并与原版本兼容。这样做既保持了原版本的优势又吸收了 Web 发展中涌现出来的新思想、新技术，始终保持系统的先进性。对于使用过原系统的设计者来说还可以大大节约学习成本。

下面先介绍框架的基础部分，一些新增添的技术将结合后面的应用来逐步讲述。

1.4 .NET 框架的基础结构

.NET Framework 的基础结构包括五大部分，它们是：

- 程序设计语言及公共语言规范(CLS)。
- 应用程序平台(ASP.NET 及 Windows 应用程序等)。
- ADO.NET 及类库。
- 公共语言运行库(CLR)。
- 程序开发环境(Visual Studio .NET)。

其结构如图 1.3 所示。.NET 框架的基础结构可以简化为图 1.4。

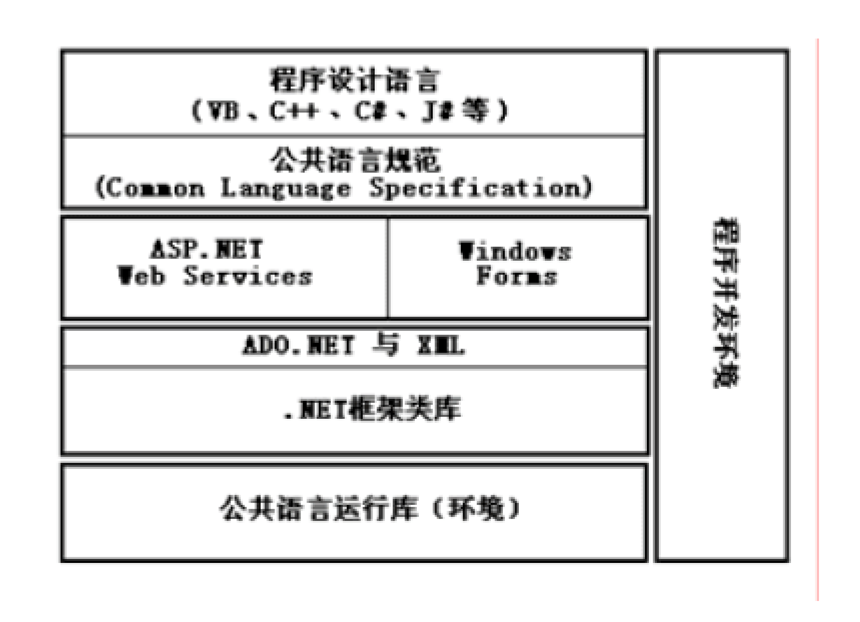


图 1.3 .NET 框架的基础结构

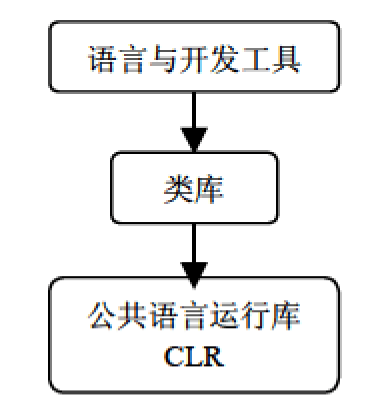


图 1.4 框架的简化图

1.4.1 .NET 框架中的程序开发语言

在.NET Framework 中可以使用多种程序开发语言，这是.NET 的一大优点。.NET Framework 中的 CLS 实际上是一种语言规范。由于.NET 框架支持多种语言，并且要在不同语言对象之间进行交互，因此就要求这些语言必须遵守一些共同的规则。公共语言规范(Common Language Specification，CLS)就定义了这些语言的共同规范，它包括了数据类型、语言构造等，同时 CLS 又被设计得足够的小。

凡是符合 CLS 规范的语言都可以在 .NET 框架上运行。目前已经有 C#.NET、VB.NET、C++.NET、J#.NET、JScript.NET 等 (VBScript 已不再使用)。除此而外，还能够运行一些现代的动态语言如 IronRuby 和 IronPython 等。

JavaScript 是各类浏览器采用的通用语言。传统的 JavaScript 是一种基于面向对象的脚本语言，现在 ASP.NET 中采用的 JScript.NET 与 JavaScript 语言完全兼容，但却已将它改造成为一种完全面向对象的语言，不仅给语言增添了很多新功能，还得到.NET 框架的完全支持。

由于多种语言都运行在.NET 框架之中，因此它们功能都基本相同，只是语法有区别。程序开发者可以选择自己习惯或爱好的语言进行开发。VB.NET 和 VC.NET 与原来的 VB、VC 相比已经有很多地方不兼容。VB.NET 和 VB 相比变化更大，VB.NET 是一种完全面向对象的语言(而 VB 只是基于面向对象的语言)。Visual J# 是.NET 框架 1.1 版本以后才增加进来的语言，用来提供给原来使用 Java 语言的程序员转向使用.NET 框架的应用程序时使用。

Visual C#.NET 是为.NET 框架“量体裁衣”开发出来的语言，非常简练和安全，最适合于在.NET 框架中使用。本书的示例都是用 C#.NET 编写的。

各种语言经过编译后，并不直接产生 CPU 可执行的代码，而是先转变为一种中间语言(Intermediate Language，IL 或 MSIL)。执行时再由公共语言运行库载入内存，通过实时解释将其转换为 CPU 可执行代码。转换的过程如图 1.5 所示。

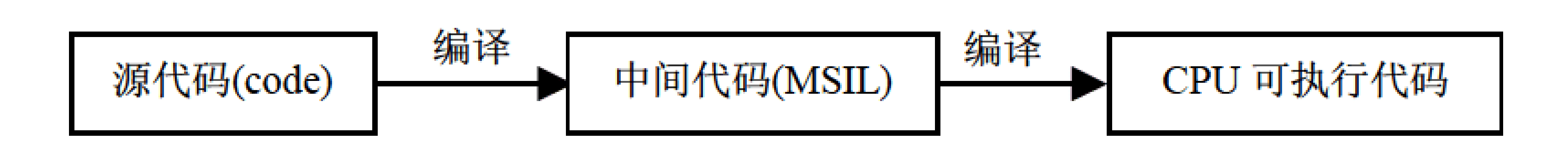


图 1.5 从源代码到 CPU 可执行代码的转换过程

设置中间语言是为了跨平台的需要。源程序经过编译转换为中间语言。各类平台只要装上不同的转换引擎，就可以将其转换为本 CPU 需要的代码[①]。由于中间语言类似于汇编语言，与二进制代码非常接近，因此实时解释的速度也很快。

转换的过程如图 1.6 所示。

---

① 到目前为止，除 Windows 以外，还没有其他操作系统安装有这样的 CLR。

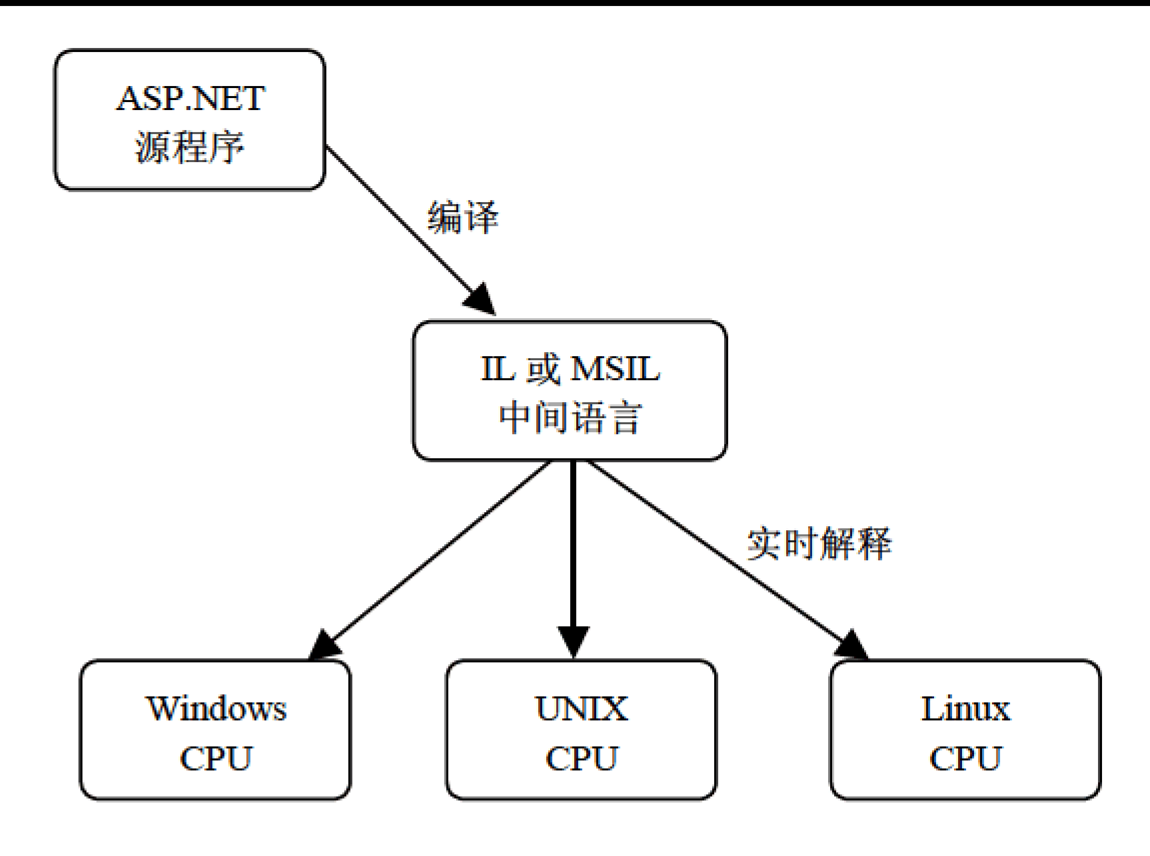


图 1.6 利用中间语言转换的过程

1.4.2 基础类库

.NET 框架的另一个主要组成部分是类库。类库相当庞大，拥有上万个可重用的“类”。各种不同的开发语言都可以用它来开发传统的命令行程序或者图形用户界面(GUI)应用程序。

.NET 框架中的类被拆分为命名空间。命名空间(Namespace)是类库的逻辑分区。类库所采用的命名空间呈层次结构，即命名空间下面又可以再分成子命名空间。每个命名空间都包含一组按照功能划分的相关的类。这样，一个大型的.NET 库就变得易于理解和便于使用。例如：

- 所有微软公司提供的类都以 System 或 Microsoft 命名空间开头。
- 有关网络协议和简单的编程接口的类放在 System.Net 命名空间中。
- 有关文件 I/O、内存 I/O、独立存储的类放在 System.IO 命名空间中。
- 基于 Windows 应用程序的用户界面的类放在 System.Windows.Forms 命名空间中。
- 有关 Web 服务器与浏览器交互，以及 Web 服务的类都放入 System.Web 及其子命名空间中。
- 所有用于处理 XML、XML 架构、XSL/T 转换、Xpath 表达式的类都放入 System.Xml 及其子命名空间中。

1.4.3 公共语言运行库(CLR)

公共语言运行库(Common Language Runtime，CLR，也称公共语言运行环境)就相当于 Java 体系中的“虚拟机”，它是 .NET 框架的核心。它提供了程序运行时的内存管理、垃圾自动回收、线程管理和远程处理以及性能优化等自动化服务。同时，它还能监视程序的运行，进行严格的安全检查和维护工作，以确保程序运行的安全、可靠以及其他形式的代码的准确性。

除此之外，CLR 还提供了客户认证、角色授权以及个性化服务等服务项目。

运行库不仅提供了多种软件服务，同时也为以往的软件提供了支持。托管和非托管代

码之间的互操作性使开发人员能够继续使用原来开发的 COM、ActiveX 控件和 DLL 等。

1.5 XML：可扩展的标记语言

当前，如何解决跨平台交换数据的问题，已经成为 Internet 进一步发展的“瓶颈”。为了解决这一问题，各个公司都曾经花费了大量的人力和物力。例如微软公司开发的 DCOM、SUN 公司开发的 CORBA 等，但都没能完全解决问题。现在 XML 的出现为解决这类问题提供了最好的机会。

1.5.1 什么是 XML

XML(eXtensible Markup Language)，是一种可以扩展的标记语言，用来描述层次化的文档。XML 是 World Wide Web(W3C)1998 年发布的标准，到现在已经发展成熟。下面先举一个简单的例子来说明什么是 XML。

一个简单的 XML 文档的结构如下。

```
<root>
   <a>
     <b>…</b>
     <c>…</c>
     <d>…</d>
   </a>
   <a>
     <b>…</b>
     <c>…</c>
     <d>…</d>
   </a>
</root>
```

一个 XML 文档必须遵循以下 5 项原则。

- 整个文档必须有，而且只能有一个“根元素”。元素严格区分大小写。
- 每个元素都是封闭的。就是说都必须有开始标记和结束标记，如果只适合用单标记时也要使用<单标记/>的方式。
- 元素之间可以嵌套，但不能交叉。
- 属性值必须包含在引号之中。
- 同一个元素的属性不能重复。

只有符合以上 5 项规则的文档才是一个具有完整结构的 XML 文稿。

另外需注意，字符“<”和“&”只能用于开始标记和引用实体。文档中可以引用的特殊字符只有 5 个：&amp、&lt、&gt、&apos 和&quot。它们分别代表“&”、“<”、“>”、“"”(双引号)和“'”(单引号)。

例如，一个“人事.xml”文档的内容如下。

```
<人事档案>
  <部门>
    <部门名>办公室
      <人员>
       <姓名>刘大为</姓名>
```

```
      <职务>办公室主任</职务>
      <职责>计划、分配、检查本部门的工作</职责>
     </人员>
     <人员>
      <姓名>李芬</姓名>
       <职务>办事员</职务>
       <职责>完成分配的工作</职责>
     </人员>
   </部门名>
   <部门名>第一车间
     <人员>
       <姓名>王自红</姓名>
       <职务>车间主任</职务>
       <职责>分配、检查本车间的工作</职责>
     </人员>
     <人员>
       <姓名>袁自立</姓名>
       <职务>钳工</职务>
       <职责>完成或超额完成生产任务</职责>
     </人员>
   </部门名>
 </部门>
</人事档案>
```

上面就是一个最简单的 XML 文档，它是一个文本文件，可以使用任何文本编辑器(如记事本等)来编写，但是.NET 提供的编写环境可以提示错误，从而给编写带来一些方便。

XML 分析器将这个文本文件转换为“文档对象模型”，形成树形层次结构。在文档对象模型中，每个标记是一个节点，所有节点必须有一个“根”(如上述程序中的<人事档案>…</人事档案>就是所有节点的根)。根的下面有若干分支，每个分支下面又可以划分出若干分支，如图 1.7 所示。

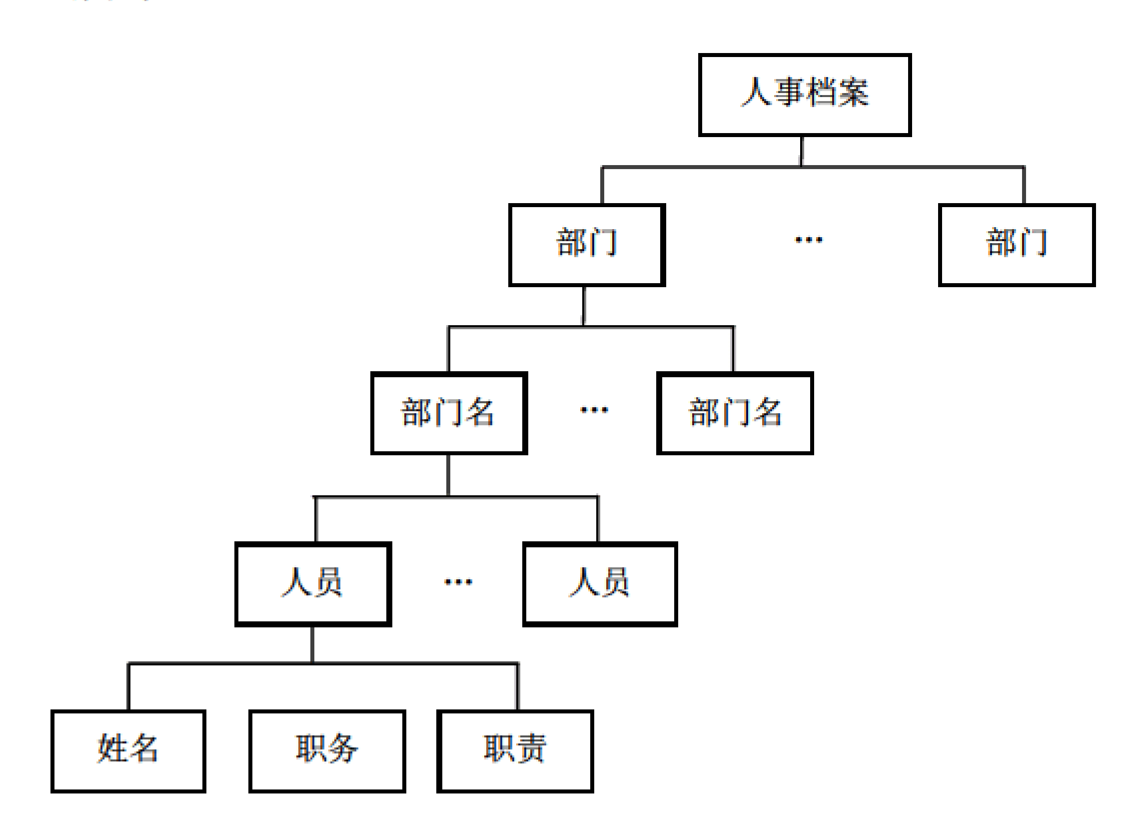


图 1.7 用 XML 文档描述的层次关系

在 HTML 中，用一套预定义的标记来格式化文本，如<html>、<head>、<h1>、<h2>、
等。为了和 HTML 区别，将其中的标记称为“元素”，在 XML 中没有预定义的元素，文档中使用的元素都是自己定义的。文档中放在“<”与“>”之间的都是元素，如人事.xml 文档中的<人事档案>、<部门>、<部门名>、<职责>、<人员>等都是元素。文

档中各元素之间存在着层次关系，下一级元素可称为“子元素”，上一级元素称为该元素的“父元素”。

在 XML 文档中，元素必须有结束标记。如“<部门>…</部门>”中“</部门>”就是元素<部门>的结束标记(元素名前加一“/”)。即使是空元素，也要用“/”结束。例如有的人没有确定的职务时，可用“<职务/>”代替“<职务></职务>”。

现在将上述文档以后缀名为.xml 的形式存储。运行该文档后在浏览器(IE 5.0 以上)中将显示如图 1.8 所示的界面。

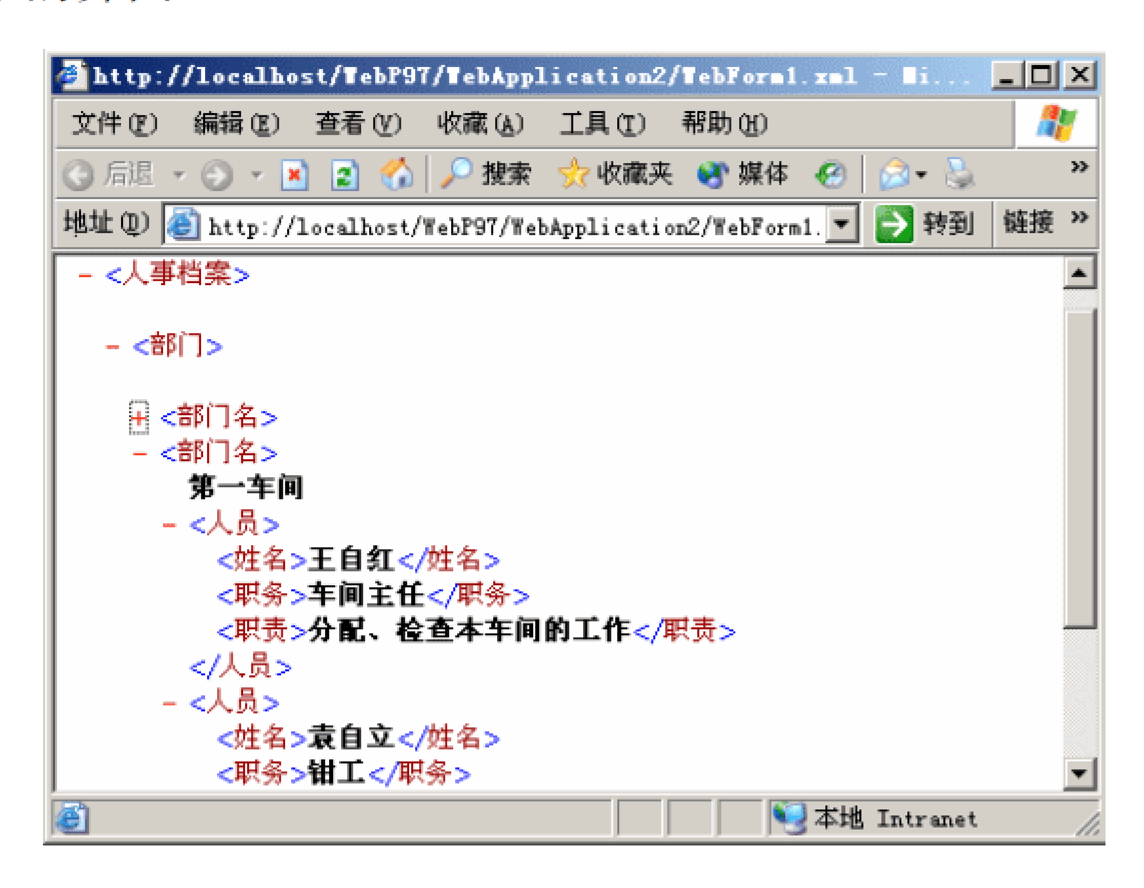


图 1.8　运行 XML 文件的结果

浏览器中显示的界面仍然保持原代码的层次结构。只是如果某个项目包括子项时，该项目名的左边多出了一个“-”号，单击这个符号时，该项目的子项将折叠在一起，“-”号变成“+”号。再单击时，各个子项将再次展开。这说明浏览器已经识别和理解了 XML 文档中的层次关系。

1.5.2　XML 的特点

归纳起来，XML 具有以下 4 方面的主要特点。

- XML 是一种通用标准，而不只是属于某个公司。不同的操作系统或工作平台，只要它支持 XML 标准，都很容易通过 XML 来交换数据。对于今天的 Web 服务器来说，XML 几乎已经无所不在了。所有的计算机平台只要获得 XML 文档，都能对文档进行分析。Windows、UNIX、Linux、MVS 和 VMS，甚至移动电话(手机)都能实现。所以 XML 文档为不同平台之间交换数据创造了极其有利的条件。
  这就好比来自不同国家的专家相聚在一起讨论问题，彼此的语言不同，如何交流？一种方法就是利用同步翻译技术，将会上的发言分别翻译成不同的语言；另一种方法就是大家都使用一种共同的语言。有人称 XML 就是网络世界的“世界语”，大家都使用 XML 时，就很好地解决了不同平台之间的信息交换的问题。
- XML 中的元素标记自行确定，不受限制，因此有很好的可扩展性。利用它几乎可以定义任何一种类型的数据，包括数学公式、软件配置说明、音乐、处方以及

财务报表等。

- XML 文档属于文本文件，语法简单，程序设计者和机器本身都能理解。可以利用任何文本编辑器编写 XML，它的语义和格式独立于平台、操作系统和应用程序。在因特网中，XML 文件可以穿过任何防火墙(因为防火墙通常不阻挡文本文件)，因而有利于数据的传输和交换。
- XML 非常有利于功能的分布。由于 XML 是对语义的描述，因此浏览器下载了服务器传来的信息后，可以自行完成信息的分类、检索、统计等操作，还为以后实现机器自动检索奠定了基础。其结果不仅大大减轻对服务器的依赖，还提高了对信息处理的效率。

当然利用 XML 文件也存在一些问题。由于文档标记自行定义，因此同样的内容可能出现几种不同的描述方式，给读者带来迷惑。为了解决这类问题，XML 常常需要结合其他几种文件一起使用，例如用 DTD 或 Schema 来定义标记，结合 HTML、CSS(级联样式表)或 XSL(可扩展样式表)来定义显示方式等。各个行业还应该根据需要定义自己的行业规范(例如 EBRL(可扩展的商业语言规范)等)。目前，XML 技术已经不单指一个文件，而是一门综合性的技术。

当在网页中创建 XML 文件时，主菜单中将出现 XML 项，选择其中的【创建架构】项时，系统将先验证 XML 文件的语法，如果有错误，将提示错误，如果没有错误时，将自动产生相应的 Schema 文件。

.NET 对于 XML 具有深层次的支持。XML 已经成为.NET 的精髓，是.NET 现在和将来发展的基础。可以将 XML 合并到数据库的记录中，Web 浏览器将接收 XML，并结合其他文件一起确定显示方法。

Visual Studio .NET 可以直接读、写 XML 文档，也可以用 XML 来描述数据的结构和系统的配置，但是用得最多的是作为数据存储和交换的格式，而这些工作对设计者来说往往是感觉不到的。

1.6 小　结

.NET 框架的基础结构由 5 个部分组成，其中最重要的是语言开发环境、类库和公共语言运行库。

在 ASP.NET 的设计中可以使用多种语言，这些语言的功能基本相同，只是语法有区别。类库中提供了数千个类，为程序设计提供了强大的支持。公共语言运行库是 .NET 框架的核心。它提供了内存管理、垃圾自动回收、线程管理和远程处理以及其他很多系统服务。除此之外，它还能监视程序的运行，进行严格的安全检查和维护工作，以确保程序运行的安全、可靠以及其他形式的代码的准确性。

将 XML 语言完全融合到自己的系统中是 ASP.NET 系统的精髓所在，虽然 XML 对数据的转换很多时候都是在幕后进行的，对设计者来说是感觉不到的，但是，还是应该知道，正是这个特点从根本上解决了不同平台之间数据的传输和转换问题，从而为 Web 服务奠定了坚实的基础。

1.7 习　　题

1. 填空题

(1) 动态网页的发展包括________、________和_________几个阶段。

(2) .NET 框架的基础结构由________、_________、_________、________和________5 部分组成。

(3) .NET 框架中包括一个庞大的类库。为了便于调用，将其中的类按照________进行逻辑分区。

2. 选择题

(1) 静态网页文件的后缀是_______。

A. .asp　　B. .aspx　　C. .htm　　D. .jsp

(2) 在.NET 中 CLS 的作用是________。

A. 存储代码　　B. 防止病毒

C. 源程序跨平台　　D. 对语言进行规范

(3) 在 ASP.NET 中源程序代码先被生成中间代码，然后再转变成各个 CPU 需要的代码，其目的是__________的需要。

A. 提高效率　　B. 保证安全

C. 源程序跨平台　　D. 易识别

(4) .NET 与 XML 紧密结合的最大好处是________。

A. 代码易于理解　　B. 跨平台传送数据

C. 减少存储空间　　D. 代码安全

3. 判断题

(1) 和 ASP 一样，ASP.NET 也是一种基于面向对象的系统。 (　　)

(2) 在 ASP.NET 中能够运行的程序语言只有 5 种。 (　　)

(3) 在内存管理中垃圾自动回收是指系统对已经不再使用的变量空间自动进行回收。 (　　)

(4) XML 中的标记由设计者自行定义，用来描述元素的内容。 (　　)

(5) XML 是一种过程语言。 (　　)

4. 简答题

(1) 静态网页与动态网页在运行时的最大区别在哪里？

(2) 试述 XML 的语法规定，并举例说明。

(3) 简述.NET 框架中 CLR 的作用。

5. 操作题

用 XML 描述班内 10 名同学 3 门功课(数学、英语和计算机)的成绩。

第 2 章　ASP.NET 网站的体系结构

为了创建 ASP.NET 网站，先要了解网站的体系结构，然后了解网站的组成以及几个重要文件的作用。本章将要讲述的问题包括:

- ASP.NET 网站的逻辑结构。
- ASP.NET 网站的组成。
- 创建新网站。

2.1　ASP.NET 网站的逻辑结构

ASP.NET 是一种完全面向对象的系统，系统建立在.NET 框架之上是它的最大特点和优点。有了.NET 框架的支持就可以快速开发出功能强大、运行可靠并且易于扩展的应用系统。

ASP.NET 网站的逻辑结构可以是两层结构也可以是三层结构。所谓两层结构是显示层直接连接到数据层；所谓三层结构是在显示层和数据层的中间增加一个商业逻辑层。两层或三层逻辑结构如图 2.1 所示。

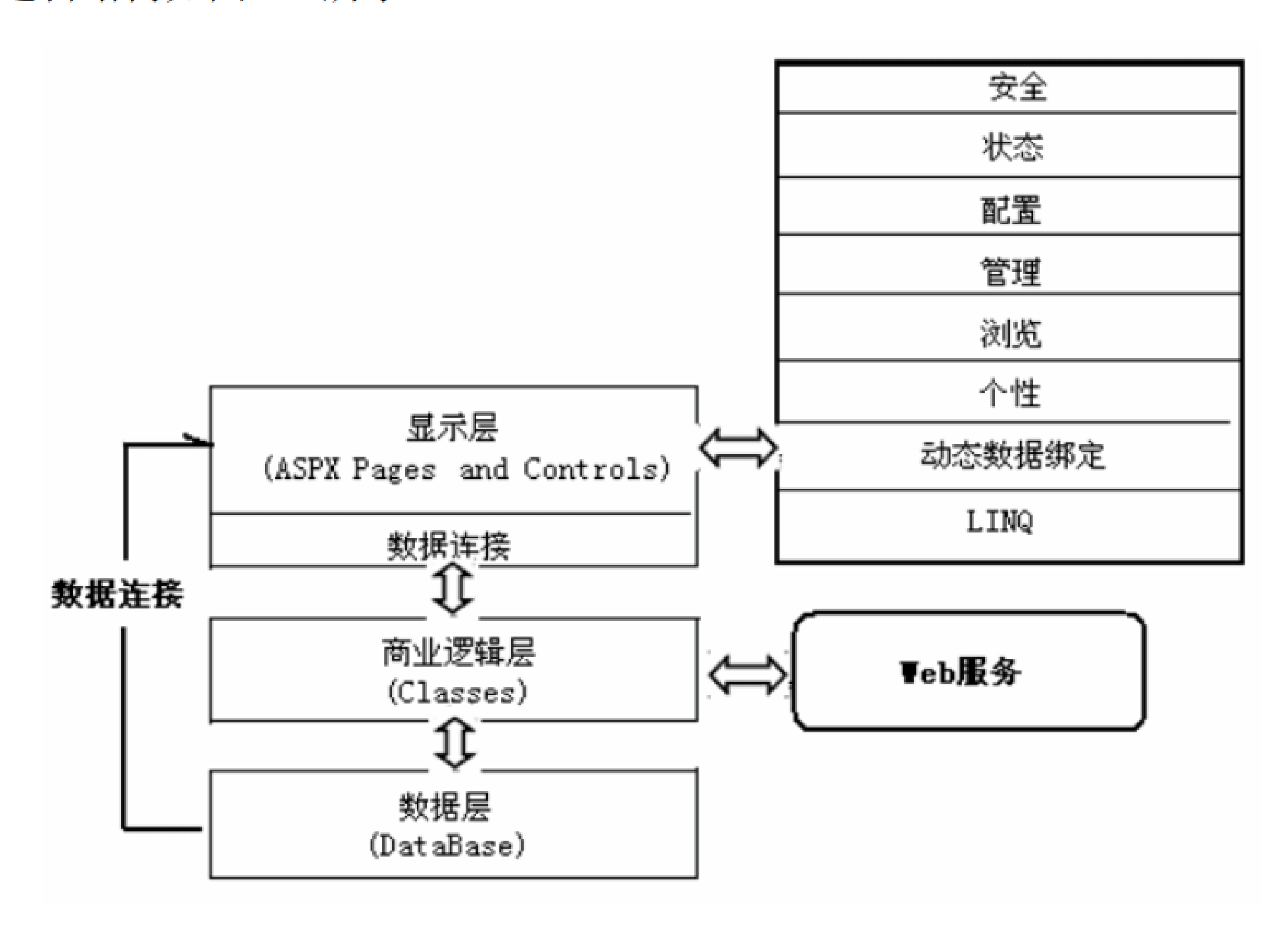


图 2.1　ASP.NET 的逻辑结构

图左边的“数据连接”代表两层结构时的连接，中间的“数据连接”代表三层结构时的连接。在三层结构中，第一层是显示层(Presentation Layer)，第二层是商业逻辑层(Business Logic Layer)，第三层是数据层(Data Layer)。如果系统比较简单时，采用两层结构比较合适。当系统比较复杂或者有特殊要求时适合采用三层结构。三层结构的中间层从

物理上看可能还包括多个层次，但从逻辑上看都属于中间层。本书的前面将重点介绍两层结构的设计方法，最后再介绍应用三层结构时的不同点。

图 2.1 的右上方列出来的是框架提供的多种服务。包括安全、状态、网站配置、网站管理、浏览、个性、动态数据绑定以及 LINQ 等，这些都是设计中经常需要涉及的部分，将其中一些成功的模式封装起来，提供给设计者调用，从而可快速地开发出功能强大而又健壮的应用系统。

图右下方的 Web 服务，是网站开发的重要手段，通过它可以借助于其他网站的支持来增强本网站的功能。

2.2 ASP.NET 网站的组成

一个网站通常由多种文件组成，主要包括以下 5 个部分。

- 一个在 IIS 信息服务器中的虚拟目录。这个虚拟目录被配置为应用程序的根目录。
- 一个或多个带.aspx 扩展名的网页文件，还允许放入若干.htm、.asp 网页或其他类型的文件。
- 零个或多个 Web.config 配置文件。
- 一个以 Global.asax 命名的全局文件。
- 几个专用的共享目录。

2.2.1 虚拟目录

虚拟目录又称为目录的“别名”，它是以服务器作为根的目录(不同于以磁盘为根的物理目录)。默认安装时，IIS 服务器被安装在“[硬盘名]:\Inetpub\wwwroot”下，该目录对应的 URL(统一资源定位符)是 http://localhost 或者 http://服务器域名。在因特网中向外发布信息或接收信息的应用程序必须放在虚拟目录或其子目录下面。系统将自动在虚拟目录下去寻找相关的文件。为了将应用程序放在虚拟目录下，有两种方法可供选择。

- 直接将网站的根目录放在虚拟目录下面。例如应用程序的根目录是 vsite，直接将它放在虚拟目录下，路径为“[硬盘名]:\Inetpub\wwwroot\vsite”。此时对应的 URL 是 http://localhost/vsite。
- 将应用程序目录放到一个物理目录下(例如 D:\site)，同时用一个虚拟目录指向该物理目录。此时客户只需要通过虚拟目录的 URL 来访问它。客户并不需要知道对应的物理目录在哪里。这样做的好处是客户无法修改该文件，一旦应用程序的物理目录有了改变时，也只需更改虚拟目录与物理目录之间的映射，无须更改虚拟目录，客户仍然可以用原来的虚拟目录来访问它们。

2.2.2 网页文件

网页(或称窗体页)是应用程序运行的主体。在 ASP.NET 中的基本网页是以.aspx 作为后缀的网页。除此之外，应用程序中还可以包括以.htm 或.asp 为后缀的网页(或其他类型

的文件)。

系统执行这些网页的内部过程是有区别的。当服务器打开后缀为.htm 的网页时，服务器将不经过任何处理就直接送往浏览器，由浏览器下载并解释执行。而打开后缀为.aspx 的网页时，则需先创建服务器控件，运行服务器端的代码，然后再将结果转换成 HTML 的代码形式送往浏览器。当然也不是每次都要在服务器端重新解读和运行，对于那些曾经请求过而又没有改变的 ASPX 网页，服务器会直接从缓冲区中取出结果而不需要再次运行。

因此，对于一个即使不包含服务器端代码的 HTML 网页，也允许使用.aspx 作为文件的后缀。此时服务器会解读此网页，当它发现其中并不包括服务器端代码时，也会将文本送往浏览器，其他什么事情也不做，其结果只是稍微降低了程序的运行效率。因此尽管允许纯 HTML 网页也使用.aspx 后缀，但并不提倡这样做。反过来，如果网页中包括有服务器控件或服务器端代码，而仍然采用.htm 后缀时，将会出现错误。

2.2.3　网站配置文件

1. Web.config 配置文件的作用

Web.config 是一个基于 XML 的配置文件，因此人和机器都能够识别。该文件的作用是对应用程序进行配置，比如规定客户的认证方法、基于角色的安全技术的策略、数据绑定的方法、远程处理对象等。其中有些问题将在以后的相关章节中讲述。

可以在网站的根目录和子目录下分别建立自己的 Web.config 文件，也可以一个 Web.config 文件都不建立。Web.config 并不是网站必备的文件。这是因为服务器有一个总的配置文件，名为 Machine.config，默认安装在“[硬盘名]:\windows\Microsoft.NET\Framework\(版本号)\CONFIG\”下。这个配置文件已经确定了所有 ASP.NET 应用程序的基本配置，通常情况下不要去修改这个文件，以免影响其他应用程序的正常运行。

在 Machine.config 与 Web.config 文件之间，以及各个目录的 Web.config 文件之间存在着一种层次关系。根目录的 Web.config 继承 Machine.config 的配置，子目录继承父目录 Web.config 的配置。只有在某个子目录的 Web.config 中有新的配置时，才自动覆盖父目录的同名配置。

2. Web.config 文件的基本结构

一个 Web.config 文件的基本结构如下：

```
<?xml version="1.0"encoding="utf-8"?>
<configuration>
  <system.web>
    <elementName1>
      <childElementName1
        attributeName1=value
        attributeName2=value
        attributeNameN=value/>
    </elementName1>
<elementName2
        attributeName1=value
        attributeName2=value
```

```
          attributeNameN=value/>
        …
      <elementNameN
        attributeName1=value
        attributeName2=value
        attributeNameN=value/>
  </System.Web>
  </configuration>
```

每个 Web.config 文件都以标准的 XML 声明开始，没有这个声明也不会出错。文件中包括<configuration>的开始标记和结束标记。

它的内部是<system.web>的开始和结束标记，表示其中的内容是 ASP.NET 特有的配置信息。这些配置信息的标记就是元素(Element)。元素可以由一个或多个子元素组成，这些子元素带有开始和结束标记，元素的内容用“名字/值”对来描述。

2.2.4 网站全局文件

Global.asax 文件(又称为 ASP.NET 应用程序文件)是一个可选的文件，一个应用程序最多只能建立一个 Global.asax 文件，而且必须放在应用程序的根目录下。这是一个全局性的文件，用来处理应用程序级别的事件，例如 Application_Start、Application_End；Session_Start、Session_End 等事件的处理代码。当打开应用程序时系统首先执行的就是这些事件处理代码。

2.2.5 几个专用的共享目录

为了管理方便，在 ASP.NET 3.5 网站中增加了几个专用的共享目录。

1. App_Code 目录

App_Code 是一个共享的目录。如果将某种文件(例如类文件)放在本目录下时，该文件就会自动成为应用程序中各个网页的共享文件。当创建三层架构时，中间层的代码将放在这个目录下以便共享。

2. App_Data 目录

如果将数据库放在 App_Data 目录下时，这些数据库将自动成为网站中各网页共享的资源。

3. App_Themes 目录

App_Themes 是一个用于放置主题的目录，在主题目录中将放入皮肤文件、样式文件和相关的图像文件，用来确定网站中各网页的显示风格。

以上所有这些共享目录的作用将结合后面的应用来讲述。

2.3　创建新网站

网站是管理应用程序并向外发布的基本单位，也是网站迁移的基本单位。在 ASP.NET 中，一个网站就是一个应用程序。由于应用的目的不同，在 ASP.NET 中可以建立三种类型的网站：文件系统网站、本地 IIS 网站和远程网站。

选择【文件】|【新建】|【网站】命令，将打开【新建网站】对话框，从中可以看见这三种网站对应的选项，如图 2.2 所示。

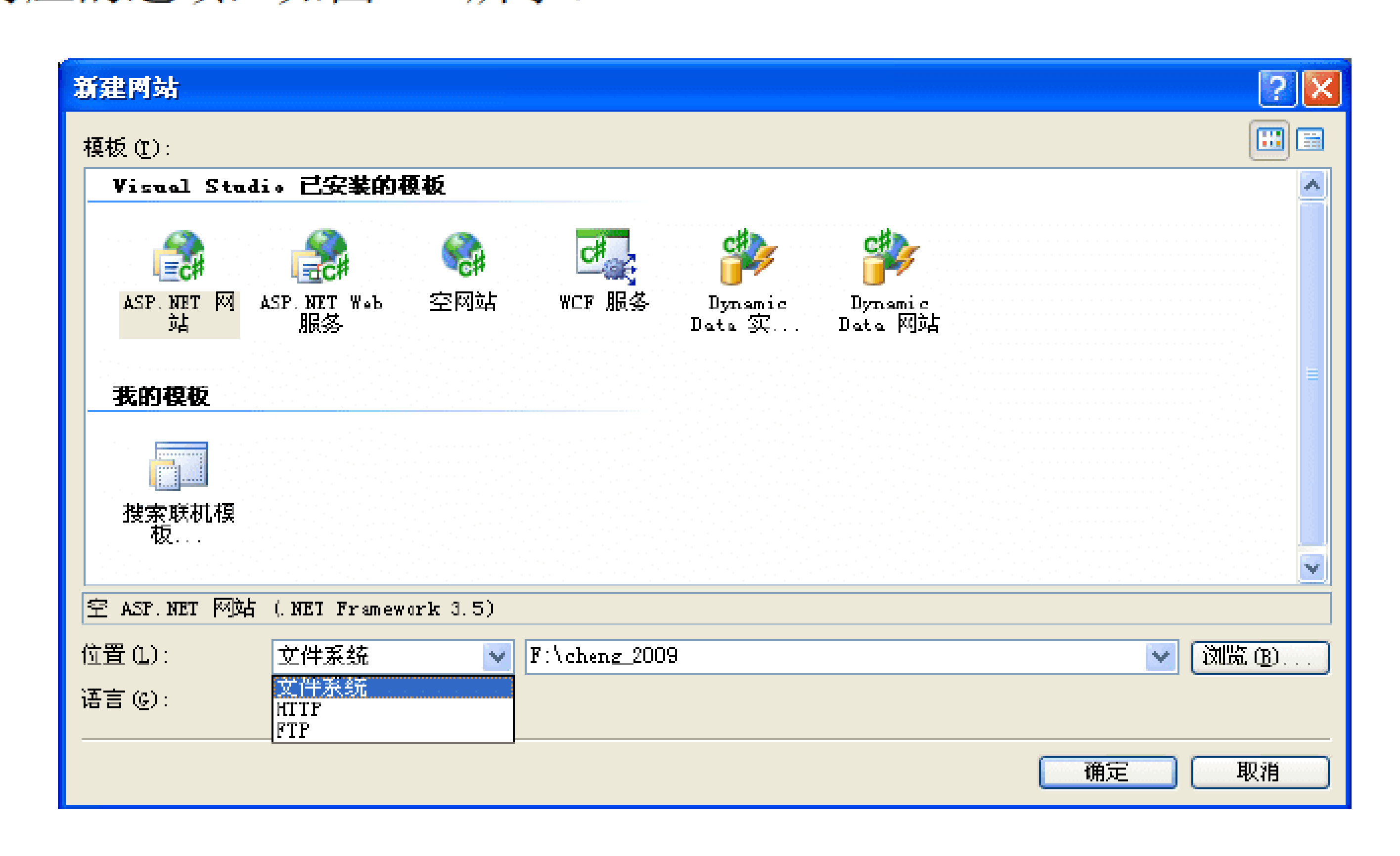


图 2.2　网站类型的选择

2.3.1　文件系统网站

文件系统网站是一种用于检查和调试的网站，只能用来检验和调试应用程序而不能向外发布信息。文件系统网站的目录可以放置在任意物理目录下面，因此非常适合于调试或者提供给学生学习时使用。

使用文件系统网站时，并不需要在计算机上安装 IIS 服务器。此时系统将自动为该网站配置一个开发服务器(Development Server)，用来模拟 IIS 服务器对网站运行时的支持。开发服务器是一种轻量级型服务器，它并不具备 IIS 的全部功能，例如它不具备邮件服务功能等。但在通常情况下，利用它进行调试已经够用。当使用文件系统网站时，系统会自动调用开发服务器来调试运行的网页，同时给网站随机地分配一个接口。例如，调试的网页名是 MyPage.aspx，当运行开发服务器时，该网页的 URL 是

```
http://localhost:31543/[网站名]/MyPage.aspx
```

其中网站名就是应用程序的根目录名。31543 在这里只是一个示例，它是开发服务器给应用程序随机生成的一个接口。

2.3.2　本地 IIS 网站

如果机器上安装有 IIS 服务器就可以创建本地 IIS 网站。此时的网站目录必须直接或

间接地放在虚拟目录下面。

创建本地 IIS 网站的步骤如下。

(1) 在打开的【新建网站】对话框的【位置】下拉列表框中选择 HTTP。

(2) 单击【浏览】按钮可以打开【选择位置】对话框，如图 2.3 所示。

(3) 在【选择位置】对话框的左边选择【本地 IIS】图标，在右上方选择两个图标之一：一个是【创建新 Web 应用程序】图标；另一个是【创建新虚拟目录】图标。前者用于直接在虚拟目录下创建网站；后者用来创建一个指向另一物理目录的虚拟目录。

(4) 如果选择【创建新虚拟目录】图标，还需要在打开的对话框中设置虚拟目录名(即别名)和对应的物理目录名，如图 2.4 所示。



图 2.3 【选择位置】对话框



图 2.4 创建新虚拟目录

本地 IIS 网站虽然提供了服务器的全套服务，但还不能向外发送信息。因为网站还不具备其他一些必要的条件，例如还没有获得唯一的 URL 的认可等。

2.3.3 远程网站

远程网站是可以向外发布信息的网站，一个远程网站必须获得唯一的 URL 地址(并且安装有扩展的 FrontPage)。为了将调试好的网站传送到远程网站，可以利用 FTP 文件服务器，将调试好的网站用字符流的方式传送到远程网站的指定目录中。为此，必须获得远程网站的允许并且取得相应的协议才可以进行此项传输工作。

2.4 小　　结

ASP.NET 是一个完全的面向对象的系统。与.NET 框架完全结合是它最大的特点，也是它最大的优点。因为.NET 框架不仅提供了庞大的类库，还提供了完善的服务，依靠这些服务可以快速创建功能强大、运行可靠的网站。

一个 ASP.NET 3.5 应用程序并不是一些孤立的网页，而是为完成一定任务的相互联系的整体，除包括多个网页以外，还需要在.NET 框架的支持下工作。网站是这个系统的管理者。离开了网站，一个单独的.aspx 网页是不能运行的(单独的.htm 网页却可以单独运行，因为它是由浏览器解释执行的)。

如果需要向外发布信息，还需要 IIS 服务器的支持，网站必须放置在虚拟目录之下。

为了使系统有效地工作，有时需要增加一些配置文件(Web.condig)、全局文件(Global.ascx)以及几个共享的目录等。

为了学习或调试程序，系统提供了一个开发服务器，这是一种轻量级服务器，使用这个服务器的最大好处是，系统不必安装 IIS 服务器，并且可以将网站放在任意物理目录之下，因而特别适合于学生学习时使用。当打开文件系统网站时，系统就会自动打开开发服务器，并且给应用程序分配一个随机产生的端口。

2.5 习　　题

1. 填空题

(1) ASP.NET 3.5 系统的两层逻辑结构适合于________的系统；三层结构适合于的系统。

(2) 默认安装中，IIS 服务器被安装在“[硬盘名]:\ ________”下。对应的 URL 是或________。

(3) 服务器有一个总的配置文件，名为________。在这个文件中已经确定了所有 ASP.NET 应用程序的基本配置。

(4) 打开文件系统网站时将自动打开一个________服务器，这是一个轻量级服务器，可以用来对程序进行检查和调试工作。

2. 选择题

(1) App_Code 目录用来放置中间层的________。

A. 专用数据库文件　　B. 共享文件

C. 被保护的文件　　D. 代码文件

(2) App_Data 目录用来放置________。

A. 共享的数据库文件　　B. 共享文件

C. 被保护的文件　　D. 代码文件

(3) 文件系统网站非常适合于学习使用，因为________。

A. 不用安装 IIS　　B. 网站允许放置在任意目录下

C. 能够进行单独调试　　D. A+B

3. 判断题

(1) Web.config 是网站中必需的配置文件。　(　　)

(2) 网站中的 Global.asax 文件(如果有的话)必须放在应用程序的根目录下。　(　　)

(3) 离开了网站，ASPX 网页能够单独运行。　(　　)

(4) 离开了 IIS 服务器，.htm 网页能够单独运行。　(　　)

4. 简答题

(1) 什么是虚拟目录？将应用程序的根目录放在虚拟目录下的方式有哪两种？

(2) 服务器打开.htm 和.aspx 两种网页时内部的执行过程有何不同？

(3) 简述 Web.config 文件的特点及文件之间的层次关系。

5. 操作题

(1) 用两种建立虚拟目录的方法实际创建本地 IIS 网站。

(2) 实际创建一个文件系统网站。

第二部分

浏览器端开发

近几年来对浏览器端的开发越来越重要，这是因为随着 Web 应用范围的扩大，参与网站的客户日益增加，特别是进入商业应用以后，客户对网页外观的要求日益苛刻，网页的质量已经成为网站间竞争的重要因素。

另一方面，浏览器本身也在发展，不仅硬件功能有所增强，软件技术也有重大突破。一些新思想、新标准和新技术的出现，解决了设计中的一些难点，减轻了浏览器对服务器的依赖程度，使浏览器与服务器之间的分工变得更加合理。

这一部分将讲述 HTML、CSS、动态 HTML 以及利用 jQuery 设计动画等四方面的内容。

第 3 章　HTML

HTML 是一种标记语言，它是网页设计的通用标准。要想设计网页，首先要学习 HTML，了解它的结构，及各标记的作用，并能根据设计的需要正确使用这些标记。

3.1　HTML 概述

HTML(Hyper Text Markup Language，超文本标记语言)是 WWW(World Wide Web)中使用的超文本标记语言，最早源于 SGML(Standard Generalized Markup Language，标准通用标记语言)，可以说它是 SGML 的一个应用程序。

HTML 是浏览器都能识别的通用语言，是浏览器生成网页的基础。利用它可以完成以下三方面的工作。

- 定义在浏览器上的显示格式。
- 建立超链接。
- 集成其他多媒体软件。

HTML 具有如下特点。

- HTML 不是一种过程语言，它只是一种标记语言，一种文本语言。它不同于我们平时所看到的 VB、C++这类编程语言，实际上它只是在一些对象(如文本、图片、表格等)中加入各种各样的标记符，从而使这些对象以这些标记符所定义的形式显示出来。
- 任何文本编辑器都可以编辑它，只要能将文件另存为 ASCII 纯文本格式即可。一个 HTML 文件是一页文字信息，就像一封电子邮件或一个 Word 字处理文件，而且实际上完全可以使用 Word 字处理软件来编写一个 HTML 网页。也可以通过其他字处理软件编写文本文件，当然使用专业的网页编辑软件更为方便一些。
- 需要使用 Web 浏览器。想把网页制作成某种模样时，需要使用一种编码通过浏览器进行解释，这种编码就被称为 HTML 代码。所有网页，都是通过浏览器对 HTML 解释而形成的，浏览器就相当于 HTML 的翻译程序，负责解释 HTML 文件各种符号的含义。一个 HTML 文件中包含了所有将显示在网页上的文字信息，其中也包括对浏览器的一些指示，如哪些文字应放置在何处，使用哪种显示模式等。如果还有一些图片、动画、声音或是任何其他形式的资源，HTML 文件也会告诉浏览器到哪里去查找这些资源，以及这些资源将放置在网页的什么位置。
- HTML 独立于各种平台。自 1990 年以来 HTML 就一直被用做 World Wide Web 的信息表示语言，用于描述网页的格式设计和它与 WWW 上其他网页的连接信息。HTML 目前已经成为各种类型浏览器的通用标准，它能独立于各种操作系统

平台(如 UNIX、Windows 等)。其内容从功能上大体可分为文本结构设置、列表建立、文本属性制定、超链接、图片和多媒体插入、对象、表格以及窗体的操作等。

3.2　HTML 标记的基础

HTML 语法格式非常简单，基本上只要明白了各种标记的用法便算学懂了 HTML。

3.2.1　基本的 HTML 语法

基本的 HTML 语法如下。

- 所有的标记都必须用尖括号(即小于号“<”和大于号“>”)括起来。它以开始标记及结束标记将元素围住。
- 大部分标记是成对出现的，包括开始标记和结束标记。开始标记和相应的结束标记定义了标记所影响的范围。结束标记与开始标记名称相同，但结束标记必须用一个斜线符号开头。
- 所有被标记包围的元素，如文本、图像、表格等都按照标记定义的格式显示。例如：

```
<h2>欢迎参观本网页！</h2>
```

  中间的字符串按照<h2>号字体的大小显示。
- 有少数标记允许只有开始标记没有结束标记。例如：断行用的标记
、分段用的标记<p>。
- 标记不区分大小写，但默认情况下，ASP.NET 中系统提供的 HTML 标记都用小写字母表示。

3.2.2　标记的属性

大多数标记都拥有一些属性，大部分属性都有默认值，利用这些属性可以对标记的作用进行更多的控制。设置或改变属性时，将属性名及其值放在标记名的同一个尖括号中。例如：

```
<p align = right>本段落将放置在右端 </p>
```

其中 p 是段落标记；align 代表“对齐方式”，是属性名；right 代表“右边”，是属性的值。这个标记的含义是将字符串放在浏览器的右端。

```
<hr size=6 width=160 align=center>
```

其中 hr 是水平线标记，size 是水平线的高度，width 是水平线的宽度，align 为对齐属性。此行代码表示增加一条横线，水平线的高度为 6，宽度为 160，显示于浏览器的中间。

3.2.3 注释语句

与其他程序设计语言一样，在 HTML 文本的适当位置增加注释语句能提高文本的可读性，编译器将不解读注释部分，即注释不在浏览器窗口中显示出来。

注释语句的格式是：

<!——注释语句——>

这里不妨将它看做一种特殊的标志。

例如：

<!——这是一条
多行的注释，
到这里结束——>

3.3 html 文档的结构

一个基本的 html 文档通常包含以下三对顶级标记。

3.3.1 html 标记

html 标记(<html>...</html>)是全部文档内容的容器，<html>是开始标记，</html>是结束标记，它们分别是网页的第一个标记和最后一个标记，其他所有 html 代码都位于这两个标记之间。html 标记告诉浏览器或其他程序：这是一个网页文档，应该按照 html 语言规则对标记进行解释。<html>...</html>标记是可选的，最好不要省略这两个标记，以保持 Web 文档结构的完整性。

3.3.2 首部标记

首部标记(<head>...</head>)用于提供与网页有关的各种信息。在首部标记中，可以使用<title>和</title>标记来指定网页的标题，使用<style>和</style>标记来定义 CSS，使用<script>和</script>标记来插入脚本等。

除此之外，在首部标记中还有一个重要的组成部分，那就是元数据(Meta)。元数据不属于文档显示内容而是用来说明文档信息的。在 html 4.0 及以后版本的浏览器中，允许用它来指定多种能被识别的说明信息。如文档的作者、时间和日期，关键字列表，使用的语言格式，网页刷新的间隔等。

这些信息有些是为了注释的需要，也有一些用于通知浏览器执行某些要求(如刷新间隔等)。例如编写的元数据如下：

```
<meta name="keyword" content="Chinese medicines" />
```

说明本网页的关键字是 Chinese medicines，它表明网页的内容与“中药”有关。一些大型网站常常利用自己的搜索引擎到其他网站去收集信息，主要就是从各网页的文档头(Head)中的关键字中去过滤、归纳相关的<meta>信息，然后在本网站上发布，以丰富本网

站的信息量，吸引更多的客户。其结果也给被搜索的网站做了宣传，对双方都带来好处。

再如编写的元数据如下：

```
<meta http-equiv="refresh" content="20" />
```

它指定了网页刷新的间隔(20 秒)。及时刷新网页，对于内容变化得比较频繁的信息来说(如股票的信息)非常重要。

3.3.3 正文标记

正文标记(<body>...</body>)包含了文档的内容，文字、图像、动画、超链接以及其他 html 元素均位于该标记中。

正文标记有下列属性。

- backgroud：指定文档背景图像的 URL 地址，图像平铺在页背景上。
- bgcolor：指定文档的背景颜色。
- text：指定文档中文本的颜色。
- link：指定文档中链接的颜色。关于链接的介绍请参阅后面章节。
- vlink：指定文档中已被访问过的链接的颜色。
- alink：指定文档中正被选中的链接的颜色。
- onload：指定文档首次加载时调用的事件处理程序。
- onunload：指定文档卸载时调用的事件处理程序。

在上述属性中，各个颜色属性的值有两种表示方法：使用颜色名称来指定，例如红色、绿色和蓝色分别用 red、green 和 blue 表示；使用十六进制格式数值#rrggbb 来表示，rr、gg 和 bb 分别表示颜色中的红、绿、蓝三基色的两位十六进制数据。

3.3.4 html 文档的基本结构

学习了文档的几个标记后，现在需要按照一定的规则组织它们才能使用。这个规则就是 html 文档的基本结构，学会它可从整体上把握 html 文档的使用。

html 文档的基本结构可以表示如下：

```
<html>
<head>
<title>标题文字</title>
</head>
<body>
文本、图像、动画、html 指令等
</body>
</html>
```

html 文本是一种树形(层次)结构。<html>标记是文本的根，其他的 html 标记全部包括在<html>…</html>以内。<html>下面有两大分支：<head>…</head>和<body>…</body>。其中<body>…</body>分支为文档的主体，主体中的内容将显示在客户端的浏览器中。<body>内又包括若干分支,如用 h1、h2 等表示字体字号，p、div、form 等表示块元素。在<head>…</head>段中除<title>…</title>包括的内容将作为窗口的标题显示在最上方外，其

余部分主要是关于文档的说明以及某些共用的脚本程序。<head>与<body>为独立的两个部分，不能互相嵌套。

3.4 html 文档的编辑工具

3.4.1 html 文档编辑器的选择

虽然任何一种文本编辑器都可以作为 html 编辑器，如 Windows 的记事本等，但为了减少网页设计的工作量，开发商设计了许多专用的网页编辑器，像 Dreamweaver、FrontPage 等，提供了可视化设计网页的工具，设计者可用“所见即所得”的直观方法来创建网页，系统再自动将其转化成 html 语句，然后传送到浏览器，从而简化了设计过程。

3.4.2 文档编辑的基本步骤

文档编辑的基本步骤如下。

(1) 创建网站，然后右击网站名，在弹出的快捷菜单中选择【添加新项】。在弹出的对话框中选择【html 页】，在左下方给网页更名，然后单击【添加】按钮。

(2) 新打开的网页左下角将出现【设计】、【源】和【拆分】三个标记，在【设计】视图中可以进行可视化设计；在【源】视图中可以直接编写 html 代码；在【拆分】视图中显示上下两部分，上方显示代码，下方显示可视化界面。当在上方书写代码时，可以立即在下方看到显示结果；在下方拖入新控件时，上方也可以立即显示出相应的代码。三个视图是等效的，可以随时切换，相互补充。

(3) 有时，在一些网页中需要多次使用某种标记，而在工具箱中找不到相应的图标时，可以先在【源】视图窗口中，写出该标记的代码，然后再将这些代码直接拖到工具箱中，以后就可以直接利用工具箱进行可视化操作了。

3.5 html 文本编辑

现在着重介绍 html 文本对象的编辑方法，掌握了这部分内容后再学习其他编辑方法也就不困难了。在这里虽然有时也要用到可视化方法，但是建议用输入 html 代码的方法试做一下以达到练习和巩固 html 的目的。

3.5.1 文本格式化

1. 字符的格式化

1) 设置字体、字号和颜色

在 html 中，可以使用字体标记…来设置文本的字符格式，为此可以将文本置于和标记之间，并通过 face、size 和 color 属性来设置文本的字体、字号和颜色。face 属性指定一种字体，或者给出一个字体列表，各种字体名称用逗号来分隔，

可以按照作者的喜好程度来排列。例如：

```
<font face = "楷体_gb2312, 仿宋_gb2312, 宋体">龙卷风</font>
```

size 属性指定字体的大小(即字号)，其取值可以从 1～7，默认值为 3。size 属性值越大，显示的字号就越大。相对于基本字体(Basefont)的大小，也可使用+或-号来指定相对字号。例如：

```
<font size = "6">龙卷风</font>
<font size = "+2">静态网页设计</font>
```

color 属性指定文本的颜色，可以用颜色名称表示，也可以用十六进制 rgb 格式表示。例如：

```
<font color = "red">龙卷风</font>
<font color = "#00ff00">静态网页设计</font>
```

2) 设置字符样式和特殊字符

(1) 设置字符样式

通过设置字符样式可以为某些字符设置特殊格式，例如粗体、斜体、下划线、删除线、上标、下标等。常用的设置字符样式的标记如下：

```
<b>…</b>粗体
<big>…</big>大字体
<i>…</i>斜体
<s>…</s>删除线
<small>…</small>小字体
<strike>…</strike>删除线
<sup>…</sup>上标
<sub>…</sub>下标
<tt>…</tt>固定宽度字体
<u>…</u>下划线
```

(2) 插入特殊字符

设计网页时，经常要插入一些空格。这本来是一个十分简单的问题，但在 html 网页中却变得比较麻烦。在输入文本时按多次空格键，但在浏览器中打开网页时却只能看到一个空格。另外，在网页中有时可能要插入一些特殊符号，如版权符号©和注册符号®等。当遇到这种情况时，可以使用两种方式来输入特殊符号：使用字符实体名称或数字表示方式。前者为&之后跟助记符，后者为&之后加#再跟十进制代码。前者的方法更为可取，因为它与平台几乎无关。例如，若要在网页中输入一个无间断空格，可以输入“ ”或“ ”。表 3.1 列出了一小部分特殊符号的实体名称和数字表示，其他可参考帮助文档。

表 3.1 部分特殊符号的实体名称和数字表示

特殊字符	代 码	字符实体名称	字符数字表示
空格	160		
©	169	©	©
®	174	®	®

2. 段落的格式化

段落是文档的基本信息单位，一个字符也可能是一个段。将文档划分为段落，可以通过使用分段标记、换行标记、标题标记或插入水平线的手段来实现。

1) 设置分段标记 p

分段标记定义了一个段落，使用该标记时要跳过一个空行，使后续内容隔一行显示。若同时使用<p>和</p>，则将段落包围起来，表示一个分段的块。若省略结束标记</p>，可以将开始标记<p>放在段尾。分段标记的常用属性是 align，用于设置段落的水平对齐方式。

2) 设置换行标记 br

标记强行规定了当前行的中断，使后续内容在下一行显示。

3) 设置标题标记 hn

标题标记用于设置文档中的标题和副标题，其中 n 的取值是 1～6。如<h1>...</h1>标记表示字号最大的标题，<h6>…</h6>标记表示字号最小的标题。

4) 设置水平线标记 hr

hr 标记在文档中添加一条水平线，用来分开文档的两个部分。该标记有以下属性。

- align：指定线的对齐方式，取值为 left(左对齐)、center(居中对齐)或 right(右对齐)，默认值为 center。
- color：指定线的颜色。
- noshade：若指定该项，则显示一条无阴影的实线。
- size：指定线的宽度，以像素为单位。
- width：指定线的长度，单位可以是像素或百分比(占页面宽度的百分比)。

5) 设置段落的对齐方式

在网页中有 4 种段落对齐方式：左对齐、右对齐、居中对齐和两端对齐。在 html 中，可以使用 align 属性来设置段落的对齐方式。align 属性可以应用于多种标记，例如分段标记<p>...</p>、标题标记<hn>...</hn>以及水平线标记<hr>等。align 属性的取值可以是 left(左对齐)、center(居中对齐)、right(右对齐)以及 justify(两端对齐)。两端对齐是指将一行中的文本在排满的情况下向左右两页边对齐，以避免在左右页边出现锯齿状。对于不同的标记，align 属性的默认值是有所不同的。对于分段标记和各个标题标记，align 属性的默认值为 left；对于水平线标记<hr>，align 属性的默认值为 center。若要将文档中的多个段落设置成相同的对齐方式，可将这些段落置于<div>和</div>标记之间组成一个节，并使用 align 属性来设置该节的对齐方式。如果要将部分文档内容设置为居中对齐，也可以将这部分内容置于<center>和</center>标记之间。

6) 示例

本示例用来显示伟大诗人李白“静夜思”的诗句。html 的代码如下：

```
<html>
<head>
  <title>html 文本的基本格式化练习</title>
</head>
<body bgcolor="green" text="white" >
<p align="center"><font face="黑体" color="red" size="12"><b><i>静 
```

```
夜 思</i></b></font></p>
<hr color="white" size="10">
<p align="center"><font face="宋体" color="#ffff00" size="15">
              床前明月光,<br>
              疑是地上霜,<br>
              举头望明月,<br>
              低头思故乡.<br></font>
</p>
</body>
</html>
```

3.5.2　列表

1. 创建有序列表

有序列表是在各列表项前面显示数字或字母的缩排列表，可以使用有序列表标记 ol 和列表项标记 li 来创建。语法格式如下：

```
<ol>
<li>列表项 1<li>列表项 2…<li>列表项 n
</ol>
```

ol 标记有两个常用属性：start 和 type。start 属性用于数字序列的起始值，可以取整数值。type 属性用于设置数字序列样式，其取值可以是如下几种。

- 1：表示阿拉伯数字 1、2、3 等，此为默认值。
- A：表示大写字母 A、B、C 等。
- a：表示小写字母 a、b、c 等。
- I：表示大写罗马数字Ⅰ、Ⅱ、Ⅲ、Ⅳ等。
- i：表示小写罗马数字 i、ii、iii、iv 等。

当位于
和
标记之间时，li 标记有两个常用属性：type 和 value。type 属性指定数字样式，其取值与 ol 的 type 属性相同；value 属性指定一个新的数字序列起始值，以获得非连续性的数字序列。

2. 创建无序列表

无序列表是一种在各列表项前面显示特殊项目符号的缩排列表，可以使用无序列表标记 ul 和列表项标记 li 来创建。语法格式如下：

```
<ul>
<li>列表项 1
<li>列表项 2
⋮
<li>列表项 n
</ul>
```

ul 标记的 type 属性用于指定列表项前面显示的项目符号，其取值可以是如下几种。

- disc：使用实心圆作为项目符号(默认值)。
- circle：使用空心圆作为项目符号。
- square：使用方块作为项目符号。

注意，在 IE 浏览器中，type 属性的值是区分大小写的。

3. 示例

无序和有序列表练习：星期一课程安排表。

```
<html xmlns="http://www.w2_org/1999/xhtml">
<head>
      <title>无序和有序列表的练习</title>
</head>
<body>
<p align="left">
       星期一上课安排表:</p>
<ul type="square">
<li>第一节:计算机导论</li>
<li>第二节:Web 应用程序</li>
<li>第三节:数据库理论</li>
<li>第四节:C#及 Windows 设计</li>
</ul>
<ol type="1" start="7">
<li>第五节:英语</li>
<li>第六节:数学</li>
<li>第七节:政治</li>
</ol>
</body>
</html>
```

同样，网页中的图像对提高生动性、增强网页吸引力方面能够发挥很大的作用，在网页的界面设计中占有重要的位置。围绕这两个问题本章将讲述以下几个问题：网页布局、插入图像和编辑图像。

3.5.3 表格和图层

1. 表格

表格可以用来组织数据用于布局，但更多的时候是，如通讯录、课程表、列车时刻表等大都采用表格的形式。将数据或图形放在表格中显得更有规律，更有利于对照和比较。

一张表格由行(Row)、列(Column)、单元格(Cell)三部分组成。创建表格实际上就是创建表格的行、列和单元格。

用 html 创建表格的方法如下。

- 创建表格的语句：<table>…</table>
- 创建表格的标题：<caption>…</caption>
- 创建行的语句：<tr>…</tr>
- 创建栏名的语句：<th>…</th>
- 创建单元格的语句：<td>…</td>

可通过 table 标记的下列属性对表格的格式进行设置。

- align：指定表格的对齐方式，取值可以是 left(默认值)、center 或 right。
- background：指定用做表格背景图片的 URL 地址。
- bgcolor：指定表格的背景颜色。

- border：指定表格边框的宽度，以像素为单位。如果省略该属性，默认值为 0。
- bordercolor：指定表格边框颜色，应与 border 属性一起使用。
- bordercolordark：指定 3D 边框的阴影颜色，应与 border 属性一起使用。
- bordercolorlight：指定 3D 边框的高亮显示颜色，与 border 属性一起使用。
- cellpadding：指定单元格内数据与单元格边框之间的间距，以像素为单位。
- cellspacing：指定单元格之间的间距，以像素为单位。
- width：指定表格的宽度，以像素或百分比为单位。

例如：

```
<table border="3" cellpadding="6" cellspacing="3">
  <caption align="top">
    <font size="5">1 次特快列车时刻表</font>
  </caption>
  <tr><th>站名</th>   <th>到站时刻</th>       <th>开车时刻</th> </tr>
  <tr><td>北京西</td>  <td> - </td>          <td>16： 0 </td>  </tr>
  <tr><td>石家庄</td>  <td> 18： 56</td>      <td>9： 08 </td>  </tr>
  <tr><td>郑州</td>    <td> 0： 01</td>       <td>0： 14 </td>  </tr>
  <tr><td>武昌</td>    <td> 6： 42</td>       <td>6： 54 </td>  </tr>
  <tr><td>岳阳</td>    <td> 9： 29 </td>      <td>9： 35 </td>  </tr>
  <tr><td>长沙</td>    <td> 11： 20 </td>     <td> --- </td>   </tr>
</table>
```

上述 html 语句对应的表格如图 3.1 所示。

一次特快列车时刻表

站名	到站时刻	开车时刻
北京西	-	16.0
石家庄	18.56	9.08
郑州	0.01	0.14
武昌	6.42	6.54
岳阳	9.29	9.35
长沙	11.20	---

图 3.1 列车时刻表

可通过 tr 标记的下列属性对指定行的格式进行设置。

- align：指定行中单元格的水平对齐方式，取值为 left(默认值)、center 或 right。
- background：给出图像文件的 URL，该图像用做指定行的背景。
- bgcolor：指定行的背景颜色。
- bordercolor：指定行的边框颜色，该属性只有当 table 标记的 border 属性取非零值时才起作用。
- bordercolordark：指定行的 3D 边框的阴影颜色，该属性只有当 table 标记的 border 属性取非零值时才起作用。
- bordercolorlight：指定行的 3D 边框的高亮颜色，该属性只有当 table 标记的 border 属性取非零值时才起作用。

- valign：指定行中单元格内容的垂直对齐方式，该属性的取值可以是 top(顶端对齐)、middle(居中对齐)、bottom(底端对齐)或 baseline(基线对齐)。

可通过 td 和 th 标记的下列属性对指定单元格的格式进行设置。

- align：指定单元格内文本的水平对齐方式，取值为 left(默认值)、center 或 right。
- background：指定图像的 URL，该图像用做单元格的背景。
- bgcolor：指定单元格的背景颜色。
- bordercolor：指定单元格的边框颜色。
- bordercolordark：用于指定单元格的 3D 边框的阴影颜色。
- bordercolorlight：用于指定单元格的 3D 边框的高亮颜色。
- colspan：指定合并单元格时一个单元格跨越的表格列数。
- nowrap：若指定该属性，则避免 Web 浏览器将单元格里的文本换行。
- rowspan：指定合并单元格时一个单元格跨越的表格行数。
- valign：指定单元格中文本的垂直对齐方式，取值可以是 top、middle(默认值)、bottom 或 baseline。

用可视化方法可以简化对表格的创建操作。利用 table 控件法的具体步骤如下。

打开【设计】视图窗口，使用模板创建表格的步骤如下。

(1) 选择网页【表】|【插入表】命令，将弹出【插入表格】对话框，如图 3.2 所示。

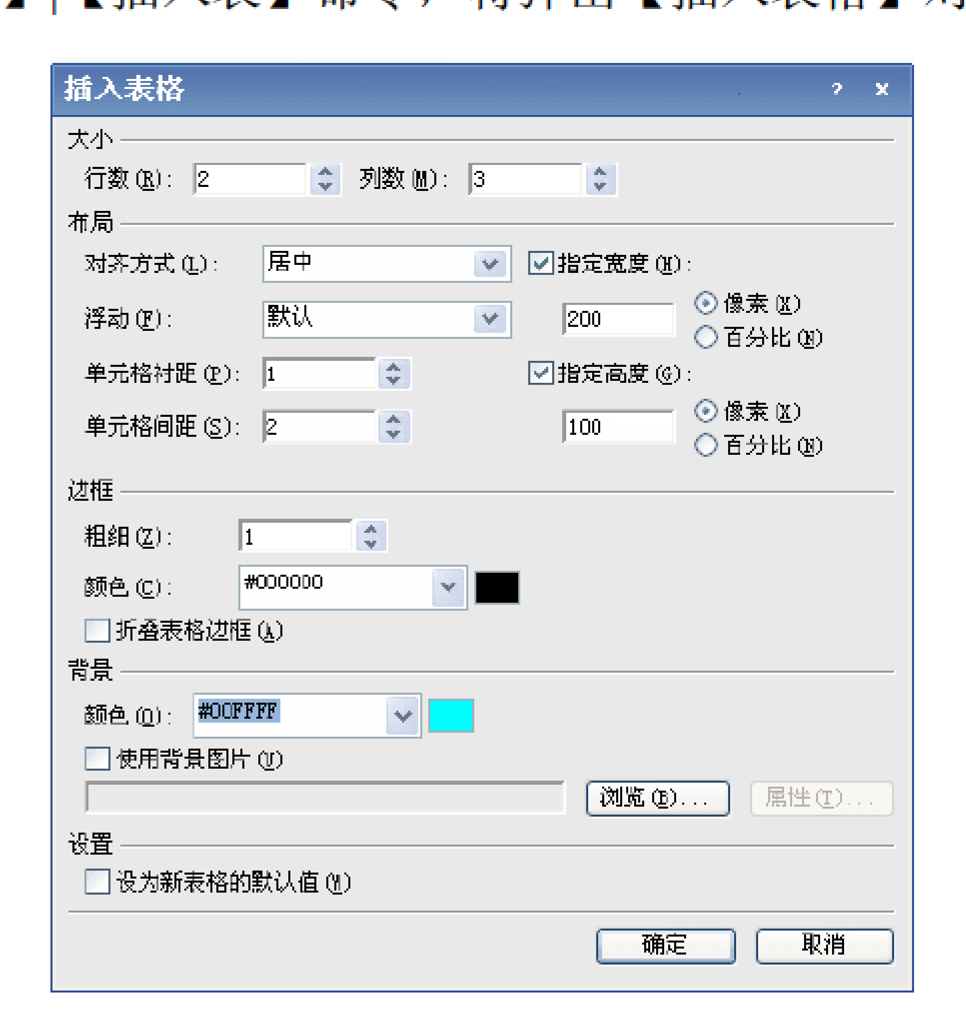


图 3.2 【插入表格】对话框

(2) 在【插入表格】对话框中分别填入需要的选项即可。按照图中的选项将创建一个两行两列，宽 200px，高 100px 居中的表格，其边框为黑色，1px 粗，表格的底色为绿色。

2. 图层

图层是一个容器，在图层内可以放置各种类型的网页元素，如文本、图像、表格，甚至还可以放置图层(图层嵌套)。每个图层相当于一个独立的小屏幕。

图层是一个可以任意移动的容器，甚至允许图层之间重叠放置，这是它与框架的不同之处，也是图层的最大优点。因为放置在图层上的元素，可以随图层被拖放到任意位置，为元素的定位和网页布局带来极大的方便，同时也为控制动态元素奠定了基础。

早期版本的 Internet Explorer 和 Netscape Navigator 浏览器都不支持图层，只有其 2.0 及以后的版本才支持图层。即使是现在，这两种浏览器对图层的定义方法似乎还没有完全统一。

在 ASP.NET 的设计界面中可以使用 div 标记来定义图层。

1) 用 html 创建图层

例如，利用下列代码创建一个宽 100px、高 100px 的图层。

```
<div style="width: 100px; position: relative; height: 100px">我是层！
</div>
```

2) 用可视化方法创建图层

用可视化方法简化了对图层的创建操作。通过鼠标的单击操作，能够创建出图层。具体步骤如下。

(1) 打开【设计】视图窗口，再将光标放置在图层将要出现的位置上，选择【工具箱】窗口的 html 选项卡，然后双击 div 控件或拖曳 div 控件到设计页面中适当位置。

(2) 选中插入的 div，利用出现的 8 个控制点，拖曳到合适的位置，并缩放为适当大小。

(3) 选中 div，在右边属性窗口中的某些属性上进行选择或输入内容，浏览时图层会发生相应变化。

3.6 插 入 图 像

3.6.1 图像的类型

并不是所有类型的图像都适合于网页的应用，有的图像虽然很美丽，但由于容量大，网上传输和下载的时间长，这样的图像就不适合网页的需要。

另外，网页中增添图像的主要目的是增加网页的生动性，使网页更具吸引力，对图像本身的精细程度要求并不高。

考虑到以上特点，PNG、GIF、JPEG 等类型的图像最适合于在网页上使用。这些图像的共同特点是：具有一定的清晰度且压缩比大、容量小，网上传输和下载的时间短。

这些图像的简要特性如下。

- PNG(Portable Network Graphic)是网络图像中的通用格式，也是 Fireworks 软件的基本格式。它用一种无损压缩的方法，最多可支持 32 位颜色，但它不支持动画，如果没有相应的插件，有的浏览器可能不支持这种格式。
- GIF(Graphics Interchange Format)是网络中最常用的图像格式。它是一种压缩的 8 位图像文件，最多可支持 256 种颜色，文件很小，适合于存储线条、图标以及卡通和其他大色块图像。GIF 图像还有一个突出的特点，就是它支持动态图、透明图和交织图。

- JPEG(Joint Photographic Experts Group，或简称为 JPG)是网络中使用频率仅次于 GIF 的图像格式。它是一种压缩得非常紧凑的格式，专用于存储不含大色块的图像。JPEG 图像有一定的失真度，但通常用肉眼不容易分辨出来。JPEG 文件最小，可以达到只有 GIF 文件的 1/4。JPEG 对图标之类或含大色块的图像不太合适，不支持透明图和动态图。

3.6.2 插入图像的方法

为了在网页上插入图像，先要将需要的图像增加到网站目录(或其子目录)中。然后在布局的基础上，利用以下两种方法插入图像。

- 将 html 选项卡中的 image 控件拖放至指定的位置，然后在它的 src 属性中设置图像的 URL。
- 在【源】视图中写以下代码。例如：

```
<img src = "images/photo01.jpg" alt = "西双版纳风光" bordor = "1">
```

用这种方法可以插入 JPG、GIF、PNG 等图像。如果要在网页中显示 Flash 动态图像时，除了需要将*.swf 图增加到网站中，还需要使用以下语句，才能将该图像显示出来。

```
<div><embed src= "frame.swf"></embed></div>
```

其中 frame.swf 是 Flash 可执行文件的名字。该文件要插入“层”(<div>…</div>)中。

3.6.3 通过属性编辑图像

在 html 中，可使用 img 标记在网页中插入一个行内图像。img 标记有许多属性，其中最常用的是 src、alt、height、width、borderhspace、border vspace 和 align 属性。而且，img 标记不仅用于在网页中插入图像，也可以用于播放 Video for Windows 的多媒体文件(.avi)，以后将分别介绍。

1. 设置图像的位置和替换文本

src 属性给出图像文件的 URL 地址，图像可以是 JPG 文件、GIF 文件或 PNG 文件。alt 属性给出图像的简单文本说明，这段文本在浏览器不能显示图像时显示出来，或图像加载时间过长时先显示出来。

2. 指定图像的高和宽

通过 height 和 width 属性来设置图像的高度和宽度，所用单位可以是像素或百分数。如果只给出了高度或宽度，则图像将按比例进行缩放。

3. 设置图像的边框

使用 img 标记的 border 属性可以给图像添加边框效果，该属性的取值为正整数，单位为像素。

4. 设置图像与文本之间的空白

使用 img 标记的 hspace 和 vspace 属性可以设置图像与文本之间的空白，前者指定图

像的左、右边距，后者指定图像的上、下边距，两者的单位均为像素。

5. 设置图像在页面上的对齐方式

如果插入的图像在页面上单独占一行，则可以将 img 标记置于<p>和</p>标记之间，并通过 p 标记的 align 属性来设置图像在页面上的对齐方式。

6. 设置图像与文本的对齐方式

当在页面中进行图文混排时，可以使用 img 标记的 align 属性来设置图像与文本在垂直方向的对齐方式，此时 align 属性的取值如下。

- top：图像与文本顶部对齐。
- middle：图像与文本中央对齐。
- bottom：图像与文本底部对齐。

通过设置 img 标记的 align 属性，也可以在图像的左、右绕排文本，此时 align 属性的取值如下。

- left：图像居左，文本居右。
- right：图像居右，文本居左。

使用换行标记 br 的 clear 属性，可以将换行后的文本移到图像的下面。

3.7 超　链　接

3.7.1 超链接的概念

1. 超链接的含义

超链接是 html 的精华。通过超链接可以随时转向其他页面或者到某个段落去查看想要看的东西，还可以跨越站点，到其他站点上去查阅相关信息。可以浏览世界各地的最新信息。超链接是由源端点到目标端点的一种跳转。源端点可以是网页中的一段文本或一幅图像等。目标端点可以是任意类型的网络资源，例如一个网页、一幅图像、一首歌曲、一段动画或一个程序等。

2. 超链接的形式

按照目标端点的不同，可以将超链接分为以下几种形式。

- 文件超链接：这种链接的目标端点是一个文件，当然包括各种文件，如页面文件。它可以位于当前网页所在的服务器上，此时可称为“与站点内页面的链接”，也可以位于其他服务器，此时可称为“与站点外页面的链接”。
- 锚点超链接：这种链接的目标端点是网页中的一个位置，通过这种链接可以从当前网页跳转到本页面或其他页面中的指定位置。
- E-mail 超链接：通过这种链接可以启动电子邮件客户端程序(如 Outlook 或 Foxmail 等)，并允许访问者向指定的地址发送邮件。
- 图像超链接：这种链接是在图像上建立若干个区域，称为“链接区”，通过它可以跳转到其他目标端点去。

3. 关于路径

路径是指从站点根文件夹或当前文件夹到目标文件所经过的路线，可以使用路径来指定超链接中目标端点的位置。在路径的表示中，常用“/”分隔文件夹，用“～”代表站点根文件夹，例如：～/image/1.gjf。路径有以下几种类型。

- 绝对路径：也称为绝对 URL，它给出目标文件的完整 URL 地址，包括传输协议在内。如果要链接的文件位于外部服务器上，则必须使用绝对路径。
- 相对路径：也称为相对 URL，是指以当前文档所在位置为起点到目标文档所经过的路径。若要将当前文档与处在同一文件夹中的另一个文档链接，或者将同一站点中不同文件夹下的文档相互链接，都可以使用相对路径，此时可以省去当前文档与目标文档完整 URL 中的相同部分，只留下不同部分。
- 根相对路径：是指从站点根目录到被链接文件的路径。使用这种路径是指定站点内文档链接的最好方式。

3.7.2 常见链接的创建

1. 创建文件链接

在 html 中，可以使用 a 标记来创建超链接。基本语法格式如下：

```
<a href = "字符串" target = "字符串" title = "字符串">文本</a>
```

上述语法格式包含 a 标记的以下基本属性。

- href：该属性是必选项，用于指定目标端点的 URL 地址，可以包含一个或多个参数。具体地，如果是与站点以外页面链接的情况，就为 URL；如果是与站点内页面之间的链接，则为文件名。
- target：该属性是可选项，用于指定一个窗口或框架的名称，目标文档将在该窗口或框架中打开。如果省略该属性，则目标文档将取代包含该超链接的文档。target 属性的取值既可以是窗口或框架的名称，也可以用_blank 指定将链接的目标文件加载到未命名的新浏览器窗口中；用_parent 指定将链接的目标文件加载到包含链接的父框架页或窗口中，如果包含链接的框架不是嵌套的，则链接的目标文件加载到整个浏览器窗口中；用_self 指定将链接的目标文件加载到链接所在的同一框架或窗口中；用_top 指定将链接的目标文件加载到整个浏览器窗口中，并由此删除所有框架。
- title：该属性也是可选项，用于指定指向超链接时所显示的标题文字。

2. 创建锚点链接

创建锚点链接时，要在页面的某处设置一个位置标记(即所谓锚点)，并给该位置指定一个名称，以便在同一页面或其他页面中引用。通过创建锚点链接，可以使超链接指向当前页面或其他页面中的指定位置。若要创建锚点链接，首先在页面中为需要跳转的位置命名，即在该位置上放置一个 a 标记并通过 name 属性为该位置指定一个名称，但不要在<a>和标记之间放置任何文字。例如，可以使用 a 标记在 test.htm 页面顶部创建一个锚点：

```
<p><a name = "top"></a></p>
```

创建锚点后，可以使用 a 标记来创建指向该锚点的超链接。例如，要在同一个页面中跳转到名为 top 的锚点处，可以使用以下 html 代码：

```
<p><a href = "#top">返回顶部</a></p>
```

若要在其他页面中跳转到该锚点，则使用以下 html 代码：

```
<p><a href = "test.htm#top">跳转到 test.htm 页的顶部</a></p>
```

3. 创建邮件链接

使用 a 标记创建邮件链接，该标记的 href 属性应由三个部分组成：第一部分是电子邮件协议名称 MAILTO；第二部分是电子邮件地址；第三部分是可选的邮件主题，其形式为“subject=主题”。第一部分与第二部分之间用冒号(:)分隔，第二部分与第三部分之间用问号(?)分隔。例如：

```
<a href = "mailto:hegels@sina.com?subject=关于动态网页设计">给我写信</a>
```

当访问者在浏览器窗口中单击邮件链接时，将会自动启动电子邮件客户端程序(例如 Outlook Express 或 Foxmail 等)，并将指定的主题填入“主题”栏中。

4. 用于下载的超链接

当需要下载某个文件时，可以利用本节中讲述的超链接方法。此方法与通常的超链接方法基本相同。只要将代码…中的 url 指向扩展名为.rar、.exe 或.dll 等类型的文件，再按照系统的提示逐步进行设置，系统将自动在指定的目录下完成文件的下载操作。

5. 创建图像超链接

图像的超链接与文本超链接差不多，就是将…标记放在图片两端即可。例如语句：

```
<a href=default.htm><img src="dysb.jpg" /></a>
```

当单击 dysb.jpg 图片时将转向 default.htm 网页。

3.7.3　示例

超链接主要形式的示例如下。(注：下面的“…”代表热点)

(1)　创建指向本地页面的链接：

```
<a href="filename.html">…</a>              //链接到本地磁盘上同一目录下的页面
<a href="../../filename.html">…</a>        //链接到本地磁盘上不同目录下的页面
```

(2)　创建指向其他服务器的页面链接：

```
<a href="http://server/path/filename.html">…</a>
(链接到 Internet 上其他服务器上的页面)
```

(3)　创建指向页面特定部分的链接：

```
<a href="#fragment">…</a>     (链接到本页面内的某指定位置)
<a href="http://server/path/filename.html #fragment">…</a>(链接到其他服务器上页面的某指定位置)
```

(4) 创建指向电子邮件的链接：

```
<a href="mailto:username@domain">…</a>
```

3.8 创建移动的文本

在网页中经常可以看到一些移动的字符串，有时称这些移动的字符串为“移动字幕”或“跑马灯”，它可以进一步吸引人们的注意力。

利用 marquee 标记可以将静态的文本转换为动态文本。该标记使用的语法如下：

```
<marquee>要滚动显示的文本信息</marquee>
```

<marquee>标记有很多属性可以改变移动的方式。

- align：指定字幕与周围文本的对齐方式，其取值可以是 top、middle 或 bottom。
- behavior：指定文本动画的类型，其取值可以是 scroll、slide 或 alternate。
- bgcolor：指定字幕的背景颜色。
- direction：指定文本的移动方向，其取值可以是 down、left、right 或 up。
- height：指定字幕的高度，以像素或百分比为单位。
- hspace：整数，指定字幕的外部边缘与浏览器窗口之间的左右边距(像素)。
- loop：指定字幕的滚动次数。
- scrollamount：整数，指定字幕文本每次移动的距离，以像素为单位。
- scrolldelay：整数，指定与前段字幕文本延迟多少 ms 后重新开始移动文本。
- vspace：整数，指定字幕的外边缘与浏览器窗口之间的上下边距(像素)。

下面举例说明这些属性的使用方法。

例 3.1 在<h2><i>移动字符串</i></h2>的前后加上<marquee>…</marquee>标志，如下：

```
<marquee><h2><i> 移动字符串 </i></h2></marquee>
```

当打开浏览器时，“移动字符串”这 5 个字将从右向左移动，到达左边沿时再快速返回到右边，继续向左边移动，不断循环直到关闭该网页时为止。根据需要也可设置成向不同的方向移动。

注意：这里虽然没有明显地用 direction 设置移动方向，但是一般每个属性都会有一个默认值，如果没有设置该属性值，就采用默认值，而 direction 的默认值为 left，即从右向左移动。如果想设置为从左向右移动，就要写上 direction=right。

例 3.2 控制移动范围。

```
<marquee width=150>再见了！</marquee>
```

用于控制移动范围。

例 3.3　单向移动。

```
<marquee behavior=slide>滑到终点了！</marquee>
```

用于控制移动到左端就停止移动。

例 3.4　左右反复移动。

```
<marquee behavior=alternate>撞来撞去，啊！我昏啦</marquee>
```

用于控制从右到左，再从左到右反复进行。

例 3.5　确定移动速度。

```
<marquee scrolldelay=15 scrollamount=50>哗！！太快了，我又昏啦</marquee>
```

结合 scrolldelay 与 scrollamount 两个属性的调整来控制每次移动的距离以及每次移动间的延迟时间，以改变移动的速度和并保证移动的平滑性。

例 3.6　用鼠标控制起停。

```
<marquee  onmouseover="this.stop()"  onmouseout="this.start()">.<div> …
</div> /marquee
```

利用鼠标覆盖或退出调用 stop()和 start()方法控制在<div>…<div>之间移动的字符串(或图片)。

3.9　HTML 与 XML 的比较

到此为止，已经介绍了两种标记语言：XML 与 HTML。它们都派生于 SGML，都是用标记来定义它们的特性，从外观上非常相似。但两者之间却有着本质的区别。

- HTML 标记是用来对 HTML 元素的外观或格式进行定义，而 XML 标记是对内容语义的描述。
- HTML 的标记由系统定义，而 XML 的标记自行定义。
- HTML 的语法比较松散，而 XML 的语法比较严格(例如，区分大小写，各标记必须封闭等)，因此 XML 的执行效率要高一些。
- 为了显示 XML 文本，常需要与其他文件结合起来使用。

当前这两种标记语言都被广泛地应用于网络应用之中。HTML 用来定义浏览器的显示，XML 主要用来在不同平台之间传输和交换数据。为了正确显示 XML 文本的内容，需要和其他文件结合，有时也需要和 HTML 相结合。

现在 ASP.NET 2.0 以及后续版本都使用 XHTML，它将 HTML 纳入 XML 的规则中。既然 XML 的标记可以自行定义，HTML 的标记当然也可以成为其中的一部分。

W3C 于 2000 年 6 月 26 日发布了 XHTML 的第一个版本作为推荐标准。它的目标是使得 XHTML 成为 HTML 的继承者。制定 XHTML 标准有两大目标。

- 文档内容与表现形式更明显地分离。
- 将 HTML 表现为 XML 应用程序。

XHTML 的标准已经发布了以下几种版本。

- XHTML 1.0 Transitional：作为从 HTML 到 XHTML 的过渡标准。
- XHTML 1.0 Strict：强迫使用 CSS 来控制外观。
- XHTML 1.0 Frameset：将浏览器划分为多个框架文档。

2001 年 5 月 31 日又发布了 XHTML 1.1 标准。它非常类似于 XHTML 1.0 Strict 版本，同时还增加了对 Mathml(数学标记语言)和 svg(可伸缩性向量语言)的支持，还可以自动创建模块元素。

目前的 ASP.NET 采用的都是 XHTML 1.0 Transitional 标准。和 HTML 相比只是语法规则严格了一些。这就是说，在 ASP.NET 中书写 XHTML 时，标记将区分大小写，各个标记必须是封闭的。例如
断行标记也应该写成
的形式。

目前是从 HTML 到 XHTML 的过渡期，在这个过渡期内，网页中的代码只要符合 HTML 4.0 要求(即使不完全符合 XHTML 的要求)，系统都会正常运行，只是用红色下划线将不符合 XHTML 标准的代码标示出来，或者在“错误列表”中发出一些错误信息的提示。

3.10 HTML 表单及其控件

3.10.1 表单(form)的作用

浏览器从服务器下载 HTML 网页以后，程序基本上按顺序运行，发生的事件都在浏览器端处理。当需要与服务器进行交互时，通过“表单”进行。

表单相当于一个容器，它把需要向服务器传送的信息收集到一起，以便一道提交给服务器进行处理。这就好比到邮局寄包裹时的“包裹单”，或到银行汇款时的“汇款单”一样，先要填写信息的传送方向、传送方式以及传送的内容等。为了便于填写这些数据，系统提供了若干 HTML 控件。

3.10.2 HTML 的表单控件

系统提供的 HTML 控件包括各种类型的输入框、下拉列表框、单选按钮、复选框等。这些 HTML 控件都是从 System.Web.UI.HtmlControls.HtmlControl 类直接或间接派生出来的类，并且都直接映射到 HTML 元素上。具体包括以下 5 种。

1. 3 种按钮

3 种按钮分别是：

- Button：一般按钮。
- Reset Button：复位按钮。
- Submit Button：提交按钮。

2. 显示控件

两种显示控件分别是：

- Label：文本显示。

- Image：图像显示。

3. 输入框

输入类型可分为：

- Text Field：文本输入或显示。
- Password Field：口令输入。
- Hidden：隐含输入。

4. 选择控件

两种选择控件分别是：

- Radio Button：单选按钮。
- Checkbox：复选框。

5. 多行显示控件

多行显示控件主要包括：

- Listbox：多行显示。
- TextArea：多行显示并可多选。

各种控件相应的 HTML 标记如下。

```
<form action="…" method="…">                    //表单开始标记
    <Input …>                                   //单行文本输入框、单选或多选按钮
    <Select …>…</Select>                        //下拉列表框
    <Textarea …> …</Textarea>                   //多行文本框
    <Input Type = "Submit" Value = "提交">      //提交按钮
    <Input Type = "Reset" Value ="复位">        //复位按钮
</form>                                         //表单结束标记
```

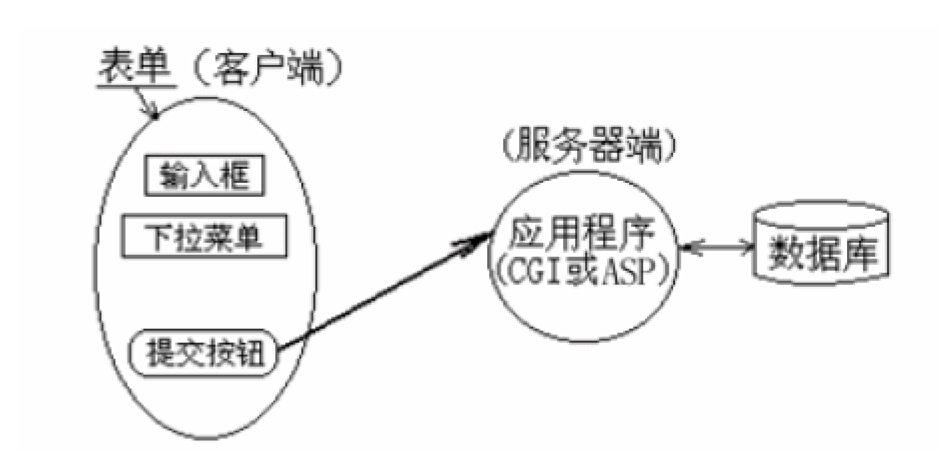


图 3.3　HTML 表单的工作过程示意

其中有两个重要的属性(action 与 method)，它们的作用如下。

- action 属性，它代表 URL，用它指向一个网页。当单击提交按钮时，立即转向该网页，以处理从表单中提交的数据。
- method 属性，该属性用来定义提交信息的方式，包括三种选择：Get、Post 和 Default。使用 Get 方式提交信息时，表单中的信息作为字符串自动附加在新的 URL 后面，立即送出，因此执行效率高，但由于 URL 的长度最长为 8K 个字符，如果提交的信息加上原 URL 超过上述长度时，超过部分将会被系统切断。使用 Post 方式提交信息时，需要对输出的信息进行包装，存入单独的文件中(不

附在 URL 后面)，等待服务器取走。利用这种方式提交时效率较低，但信息量几乎没有限制。使用 Default 方式时，提交信息的方式取决于浏览器的设置。通常情况下为 Get 方式。

其中有两个专用按钮为：【提交】按钮与【复位】按钮。系统已经为它们编写好了脚本，可用来完成以下专门的任务。

- 【提交】按钮(Submit)用来向服务器提交表单元素中的数据。
- 【复位】按钮(Reset)用来将表单元素中的数据复位(清空)，以便重新输入新数据。

当客户在这些输入控件中输入数据后，单击提交按钮，将信息一起传送到服务器，并转向另一个网页进行处理，并将处理的结果显示在另一个网页中。

每一张网页中可以包括多个表单，每个表单最少应有一个【提交】按钮和一个【复位】按钮。

3.10.3 表单示例

表单的示例如图 3.4 所示。

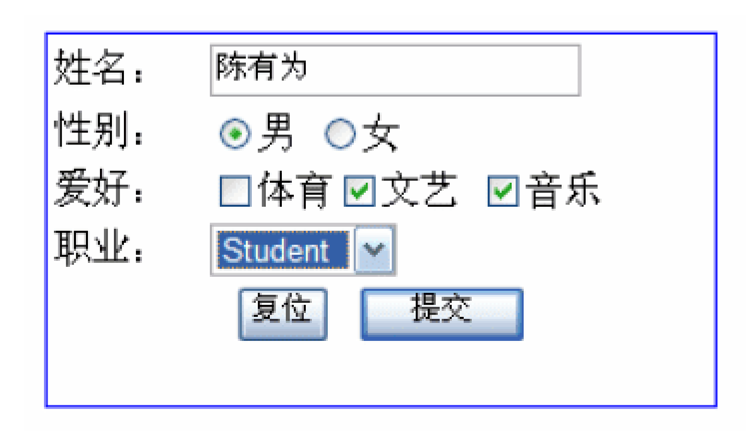


图 3.4 表单界面

其代码如下。

```
<form action="Default.aspx" method="get">
  <table class="style2">
    <tr>
        <td>
            <span lang="zh-cn">姓名：</span></td>
        <td>
            <input id="Text1" type="text" name="name1"/></td>
    </tr>

    <tr>
        <td>
            <span lang="zh-cn">性别：</span></td>
        <td>
            <input id="Radio1" type="radio" checked="checked" value="man"
            name="sex"/>男
            <input id="Radio2" type="radio" value="woman" name="sex" />女
</td>
    </tr>
    <tr>
        <td>
            <span lang="zh-cn">爱好：</span></td>
```

```
        <td>
            <input id="Checkbox1" type="checkbox" name="ty" />体育
            <input id="Checkbox2" type="checkbox" name="wy" />文艺
            <input id="Checkbox3" type="checkbox" name="yy" />音乐</td>
    </tr>
    <tr>
        <td>
            <span lang="zh-cn">职业：</span></td>
        <td>
            <select id="Select1" name="zy">
                <option>Teacher</option>
                <option>Engineer</option>
                <option>Student</option>
            </select></td>
    </tr>
    <tr>
        <td align="center" colspan="2">
            <input id="Reset1" type="reset" value="复位"
            />  <input id="Submit1"
                type="submit" value="提交" class="style3" /></td>
    </tr>
   </table>
</form>
```

如果表单的属性是 method="get"，则当填完表单中的数据并单击提交按钮时，填写的数据将附在新 URL 后面。格式如下。

```
http://www.域名/Default.aspx?name1=value1&name2=value2…
```

问号后面就是表单中填写的数据，格式采用“名称=值”的形式，语句中的空格一律自动用&取代。

如果 method="post"，则需要传出的数据将另外打包，不附在新 URL 后面。

作为处理信息的网页(本例中为 Default.aspx)通常需将信息存入数据库中并向客户反馈一定的信息。关于存入数据库的方法将在后面章节中讲述，这里只演示反馈信息。为此在处理信息网页的 Page_Load 事件中写出以下代码：

```
protected void Page_Load(object sender, EventArgs e)
        {
          string tt = Request["name"];                // 取出输入的名字
          Response.Write(tt+"先生，感谢您登录网站！");    // 反馈信息
        }
```

3.11　小　　结

一个好的网页中常常包含各种不同的文本，这些文本的大小不同，形式也不相同，是什么方法使它们变得如此丰富多彩呢？是 html(加上下一章将介绍的 CSS)。

在桌面系统中，利用文字编辑软件(例如 Word)编辑文本时，将文本写成什么样将来就显示成什么样。但是浏览器与此不同，它只认识 html。只有用标记才能定义显示的形式。

为了显示不同的形式，系统确定了不同的 html 标记。使用时只需要将这些标记把定

义的对象包围起来即可。由于使用方法非常简单，因此受到广泛的欢迎，html 已经成为各种类型的浏览器的通用语言。

标记中的属性如宽度、高度、颜色以及对齐方式等，作为标记定义的补充，大大增强了标记定义的表现力。

文本或图像超链接是 html 的灵魂，正是因为有了它，才可以让人们在网上冲浪，遨游世界。在 html 中超链接定义的最简单的格式是

```
<a href = "url" >链接文本</a>
```

动态文本常常能够起到突出重点、吸引更多网民目光的效果。定义移动字符串的语句是

```
<marquee>移动的文本</marquee>
```

移动图片的定义语句与此相同。格式如下。

```
<marquee>移动的图片</marquee>
```

即在<marquee>的标记中间放进一张或多张图片，这些图片就可以用移动的方式显示。

在<marquee>标记中还可以增加很多属性，用来控制移动的方式(如向左、向右、往返、起停等)和移动的速度。

浏览器从服务器下载.htm 网页以后，程序基本上都在浏览器端运行。当需要与服务器进行交互时，通过表单进行。表单负责将信息收集到一起，以便一道提交给服务器。在表单中 action 属性指向处理信息的网页，method 属性用来指定信息传输的方式。传输方式包括 post 和 get 两种。

3.12 习 题

1. 填空题

(1) HTML 是__________的英文缩写。它是 WWW(英文为________)中使用的超文本标记语言。它最早源于 SGML。

(2) 对于 HTML，任何________编辑器都可以编辑它。它目前已经成为各种类型______的通用标准。

(3) 所有网页，都是由_______对 html 解释而形成的，________就相当于 HTML 的翻译程序，负责解释 html 文件各种符号的含义。

(4) 列表文本有________和__________两种形式。

2. 选择题

(1) 在网站中，路径通常有_______种表示方式，它们分别是_______。

A. 3，绝对路径、根目录相对路径、文档目录相对路径

B. 2，绝对路径、根目录相对路径

C. 3，绝对路径、根目录绝对路径、文档目录相对路径

D. 2，绝对路径、根目录绝对路径

(2) 在 html 中，超链接由________标记定义。

A. <p>　B. <a>　C. <img>　D. <meta>

(3) 下面关于绝对路径的说法，正确的是________。

A. 绝对路径是被链接文档的完整 URL，不包括使用的传输协议

B. 使用绝对路径需要考虑源文件的位置

C. 在绝对路径中，如果目标文件被移动，则链接同样可用

D. 创建外部链接时，必须使用绝对路径

(4) html 代码 表示________。

A. 创建一个超链接

B. 创建一个自动发送电子邮件的链接

C. 创建一个位于文档内部的链接点

D. 创建一个指向位于文档内部的锚点

(5) 表单中的属性 action 是指________，属性 method 是指________。

A. 处理信息的事件

B. 传送信息的方式

C. 处理信息网页的 URL

D. 处理信息的方法

3. 判断题

(1) HTML 是对显示的描述，而 XML 不仅能描述显示还能描述语义。　(　)

(2) HTML 与 XML 一样，它们的标记都是严格区分大小写的。　(　)

4. 简答题

(1) HTML 页面的基本结构是怎样的?

(2) 在 HTML 页面中如何实现左对齐、右对齐和居中对齐?

(3) 如何在页面中插入锚点? 说出好处。

(4) 是否可以在不同的页面间实现用锚点链接进行跳转? 如果可以，怎样实现?

(5) 简述 HTML 与 XML 之间的区别。

(6) 表单的作用是什么?

(7) 从表单向服务器传送信息的方式有几种? 有什么区别?

5. 操作题

在网站中创建多个.htm 网页，要求在网页中实现以下技术。

(1) 使用各种不同字体、颜色的文本。

(2) 使用两种列表。

(3) 超链接(文本超链接、锚点超链接、电子邮件超链接等)。

(4) 结合移动字符串进行超链接。

第 4 章　CSS

随着网站应用范围日益广泛，对网页设计的要求越来越高。一张好的网页既要有鲜明的主题、丰富的内容，还要有美丽的外观和强烈的吸引力。在这种情况下，单纯利用 HTML 及其属性来设计外观的方法已经远远不能满足需要。CSS 技术的出现弥补了上述技术的不足。当前，利用 CSS 设计网页外观已经成为最佳方式。这不仅因为 CSS 具有很强的表现力，还因为它能使网页内容与显示代码分离，从而有利于代码的重用，因此用 CSS 设计网页已经成为新一代网页设计的共同标准。

ASP.NET 3.5 在 ASP.NET 2.0 的基础上增强了对 CSS 技术的支持，增加了几个处理 CSS 的工具，如应用样式窗口、管理样式窗口、CSS 属性窗口以及样式应用工具栏等，利用这些工具可以简化 CSS 的设计过程。

下面将分两阶段来介绍 CSS 技术。先讲述 CSS 的一般原理，然后结合 ASP.NET 3.5 提供的工具来讲解用 CSS 设计网页的方法。具体内容包括:

- CSS 的基本概念。
- CSS 的定义方法。
- CSS 网页布局。

4.1　CSS 的基本概念

4.1.1　什么是 CSS

CSS(Cascading Style Sheet，级联样式表)就是一组用来控制网页元素外观的属性。它由 W3C(World Wide Web Consortium)组织创建，应用于 Netscape 和 Internet Explorer 等浏览器 4.0 以及以后的版本中(3.0 版本只能部分支持 CSS 技术)，目前已为各种类型的浏览器接受，实际上已经成为业界共同的使用标准。

HTML 是一种标记语言，虽然它本身已经规定了浏览器中网页元素的显示格式，但是随着 Internet 应用范围的扩大，特别是动态网页的出现，已暴露出它的严重不足。比如：

- 它将内容和属性(用来确定显示)紧紧捆绑在一起造成了代码文件的混乱，并大大减少了代码的可重用度。
- HTML 标记的定义等级少、跨度大，很难进行微量调整。
- HTML 给元素定位的能力差，只能利用表格进行粗略定位等。

4.1.2　CSS 的作用

CSS 是对 HTML 功能的扩展。用 CSS 可以控制大多数传统的文本属性，如字体、字形、字号、字距等，还可以控制如页边距、缩排、颜色、底图等布局属性。使用 CSS 不光可以控制某些指定的 HTML 标记，还可以一次控制一篇或多篇文档的格式。

CSS 的作用可以归纳如下。

- CSS 能对网页进行“精细加工”，以美化由 HTML 初步定义的网页。
- 一个 CSS 样式表可以定义多个 HTML 文本和图像，只要改变 CSS 文件，多个 HTML 文档和图像的外观都将随之改变。反过来，同一个 HTML 文本、图像用不同的 CSS 样式表来定义，可以显示出不同的风格。CSS 使得 HTML 的文本和图像的内容与显示风格分离，从而增强了代码文件的可重用度。就好比一个人(比作内容)与他所穿的衣服(比作外观)是两件相关而又不同的实体，将它们紧紧捆绑在一起是不恰当的，只有将两者分离才能释放出更多的灵活性。
- 如果多种 CSS 的定义之间出现矛盾时，系统的处理原则是：定义的范围越小时，它的优先级越高。这也就是级联样式表中“级联”的含义，级联在这里就是“优先级”的意思。
- CSS 提供的属性能用脚本语言控制，为实现动态网页奠定了基础。
- CSS 与 XML 结合能表现出更为复杂的样式。

4.2　CSS 的定义方法

4.2.1　两种定义方式

CSS 的定义方式有两种：内联方式与链接方式。使用内联方式时，将 CSS 的定义直接写在 HTML 标记的属性中；使用链接方式时，CSS 的定义集中放置在外面的文件(.css 文件)中，或者放在本网页的某个地方，HTML 标记则通过某种属性(这里称为“选择器”)与这些 CSS 定义连接在一起。

链接方式是 CSS 的主要定义方式，因为这种方式能够将内容与表现代码分离，有利于代码重用，还可以使多张网页的显示风格一致。

但有时也需要使用内联方式，这种方式的优点是比较直观，有些特殊情况也需要使用它。例如，网站中多张网页已经用链接方式连接到 CSS 文件中，但其中某张网页的某些局部有不同的外观要求时，可以为该 HTML 标记添加 CSS 内联定义。因为 CSS 内联定义的优先级高于链接方式。

4.2.2　定义语句

1. 内联方式时的定义语句

在各种 HTML 标记中都有一个 style 属性，这个属性其实就是 CSS。可以直接用手写或者可视化方式(后面讲述)将属性写入 style 属性中。例如：

```
<h2 style="background-color:Red">利用级联样式表</h2>
```

代表将这个<h2>…</h2>标记以内字符串的背景颜色定为红色。

2. 链接方式时的定义语句

用链接方式定义时需要按照以下格式定义：

```
<head>
<style type="text/css">
选择器名 1{
    属性名 1  : 值 1;
    属性名 2  : 值 2;
    属性名 3  : 值 3;
    …
}

选择器名 2{
    属性名 1  : 值 1;
    属性名 2  : 值 2;
    属性名 3  : 值 3;
    …
}
…
</style>
</head>
```

CSS 的选择器主要包括以下几种。

1) 标记选择器

使用标记选择器的作用，实际上是对 HTML“元素”进行重定义。其格式如下。

```
HTML 标记名{  /* 显示风格的描述语句 */ }
```

如：

```
h1 {
font-size : 12px;
color : #FF0000;
}
```

即：网页中在标记<h1>…</h1>中间的字符都用 12px 大小，红色(#FF0000)来显示。

2) id 选择器

用 id 选择器时，先需要在“元素 id”名的前面加上＃号，再定义显示的风格。其格式如下。

```
# id 名 {  /* 显示风格的描述语句 */ }
```

例如：

```
#Content {
          position : absolute;
          top: 0;  left : 0;  width : 10em;
          }
```

引用示例：

```
<div id=" Content " >…</div>
```

在这里<div>…</div>标记代表“层”元素，它定义了一定的范围，在这个范围内的元

素起点用坐标的绝对值定位，它与外框之间的上边距及左边距皆为 0，其宽度为 10em(em 代表当前的字符宽度)，高度没有定义。

3)　类选择器

使用类选择器时，先需要在类名前面加上句号(.)，再定义显示的格式。格式如下。

```
.类名 {  /* 显示风格的描述语句 */ }
```

例如：

```
.details {
          color : #FF0000;
          }
```

类名定义可以应用于任意的 HTML 标记中，只要在该标记的属性中注明了类名即可。例如：

```
<p class="details">…</p>
```

上述定义表明在<p>与</p>标记之间的元素将按照.details 类名的定义来显示，即文本呈红色。

如果在同一个标记的属性中既引用了“类名”又引用了“元素 id”，而且两者的定义有冲突时，元素 id 的优先级将高于类名定义。这就是说，此种情况下，将采用元素 id 的定义。

4)　后代选择器

由于在 DOM(将在第 5 章中讲授)中的元素为层次结构，可以给某元素的后代中某些元素定义 CSS。例如：

```
.link a {
        color : blue;
        }
```

表明类名为 link 元素的后代中的 a 标记中的文本颜色为 blue。

5)　群组选择器

如果一组 HTML 标记拥有相同的 CSS 定义时，可以采用群组选择器。例如：

```
td, p, div a{
             font-size : 12px;
             }
```

表明 td、p、div a(div 标记后代中的 a 标记)三种标记中字符的大小都是 12px。

6)　通配选择器

当文档中所有元素使用同一种 CSS 定义时，可以用通配选择器“*”。例如：

```
*{
     font-size:14px;
 }
```

从上面的示例可以看出，用选择器匹配的元素并不只是单一的元素，而是一组元素的集合，凡是符合选择器名的(不只是某一个元素)都将按照该 CSS 的定义来显示外观。

> **提示：** 定义 CSS 的语法虽然简单，但是每种选择器的“属性名”和“值”较多，不易记忆，也容易写错。下列方法可以帮助我们减少错误。即当写完
> ```
> <style type="text/css">
> 选择器名{
> ```
> 后按 Enter 键，再按空格键时，系统将提示该选择器所有的属性名，选择其中之一后再按冒号(:)键，将提示出该属性中所有可选择的“值”。

3. 链接外接 CSS 文件的方法

如果使用外接 CSS 文件时，先要建立一个以.css 为后缀的文件。文件中包括若干 CSS 的定义语句。例如：

```
选择器名{
        属性名 1  : 值 1;
        属性名 2  : 值 2;
        属性名 3  : 值 3;
        …
        }
```

然后在调用该 CSS 文件的网页中书写以下语句。

```
<head>
    …
<link rel="stylesheet" type="text/css" href="CSS 的文件名.css" />
</head>
```

若要在整个网站中引用某个 CSS 文件时，可将 CSS 文件与皮肤文件一起放在主题目录下，并在网站的 Web.config 文件中进行配置。此问题将在第 19 章中进一步讲述。

4.3 CSS 网页布局

4.3.1 概述

网页的整体布局非常重要，这是基于两方面的原因：一方面网页的整体布局，将给人带来整体视觉上的直接感受；另一方面网页布局与一般的文本编辑不同，如果方法用得不当时，常常会出现在设计阶段似乎较好的界面，一旦运行布局就乱了的情况。

传统的布局方法是利用表格进行，利用表格的定位、对齐等功能将各种网页元素放置到表格指定的单元格中。虽然表格的大小可以任意调整，行、列之间也可以任意拆分或归并，但是在布局中仍然会出现顾此失彼的感觉。要想成为一名布局高手，非常不容易。

新一代的网页布局是以 DIV+CSS 为主并结合表格定位的方法。为此首先需要掌握 CSS 与布局相关的一些基本概念。

4.3.2 网页中的框架模型

网页中有些元素是其他元素的容器，如 div、body、p、h1 或其他输入控件等，这些

控件本身就好比是一个箱体，将它们称为框架模型(或简称为方框)。

1. 框架模型的参数

框架模型是 CSS 布局的基石之一。框架模型是一个矩形框，由元素内容(Content)、填充(Padding)、边框(Border)和空白边(Margin)等组成，其中 Margin、Padding、Border 又可细分为 top、right、bottom、left 4 个方向，如图 4.1 所示。

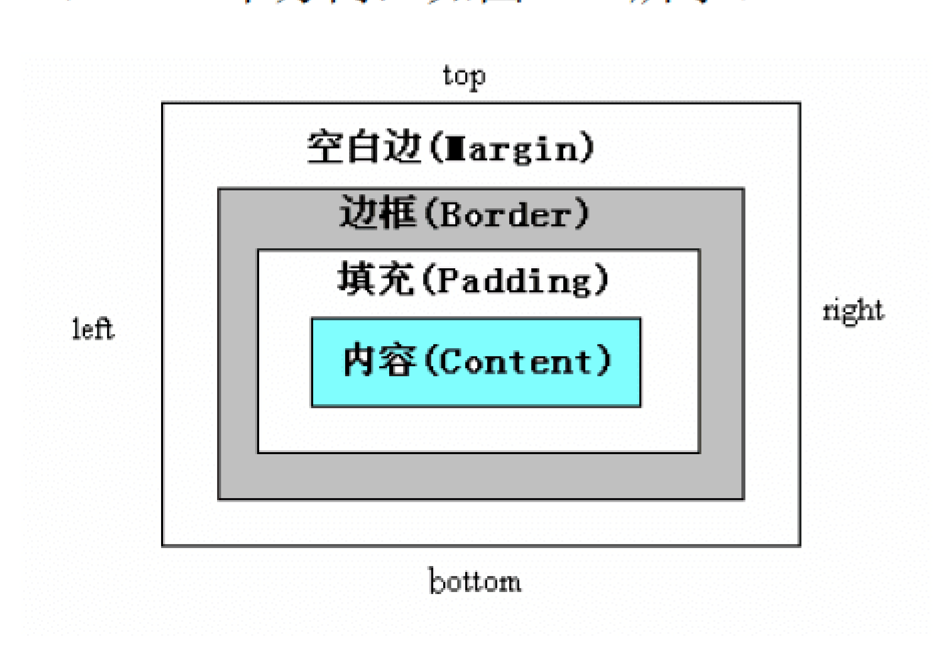


图 4.1　框架模型

网页布局实际上就如同在仓库中摆放各种箱体一样，要确定箱体的摆放顺序，边框的厚度、边框内的间隙、边框外留的空间等。

提示： CSS 中所谓宽度是指 Content 的宽度。而 IE 4.0 及以前的版本与此定义稍有区别，它所指的宽度是左右边框外沿之间的宽度。IE 7.0 版本与 CSS 的定义取得了一致。

2. 框架的长度单位

框架模型中使用的长度单位主要有三种：像素(px)、字体宽度(em)、百分数(%)。三种单位定义长度时各有利弊。

- 用像素作为单位：像素是与屏幕分辨率有关的单位。当屏幕的分辨率改变时，长度也会作相应的变化。
- 用字体宽度作为单位：这是一种与字号相关的定义。使用默认字号时，1em 是 10px，而 1 个默认字号大约相当于 16px，所以 1em 的大小相应于 62.5%字号。同样的设置，当字号大小改变时，显示的长度将跟随变化，使得字号与各字的间隔也同步协调地改变。
- 用百分比作为单位：表示本框架占有其外框架的比例，比如将本框架的宽度设为 25%时，代表本框架占据外框架宽度的 1/4。当外框架大小改变时，本框架的宽度也将自动变化，但占据的比例不变。这种设置能够适应屏幕大小的变化，这是它的最大优点。但是当浏览器窗口变得太大或太小时也会影响客户的阅读，此时可以设置它的最大宽度和最小宽度(IE 5.0 或 IE 6.0 不支持这个功能)。

3. 对框架定义的示例

下面举例说明对框架定义的方法。

(1) 以 px 作为单位时的定义：

```
p {
        border-top-width: 1px;
        border-right-width: 5px;
        border-bottom-width: 10px;
        border-left-width: 20px;
      }
```

代表 P 元素框架的上边框宽度为 1px，右边框宽度为 5px，下边框宽度为 10px，左边框宽度为 20px。

上述定义也可简化为

```
p {
        border-width: 1px 5px 10px 20px;
     }
```

注意，定义的顺序是上、右、下、左，顺时针方向。后面均与此相同。

(2) 以 em 作为单位时的定义：

```
p {
    padding : 1em 2em 3em 4em;
   }
```

这里 P 元素的上、右、下、左的填充分别设置为 1、2、3、4 个 em 单位。

有时定义中包括缺项。例如：

```
p {
    padding : 1em 2em;
   }
```

代表框架的填充的上、下为 1em，左右为 2em。

如果定义的格式如下：

```
p {
    padding : 1em 2em 3em;
   }
```

则表示上方的填充为 1em，左右填充为 2em，下面的填充为 3em。

(3) 用百分比作为单位时的定义：

```
#wrapper { width: 85%; }
```

将本框架(id ="wrapper")的宽度设置成外框的 85%。

4. 框架的排列方式

在布局中框架的排列方式主要有三种：普通流(Float)、相对定位(Relative)和绝对定位(Absolute)。

1) 普通流

普通流是默认情况下元素的排列方式。即各个元素在外框容器的范围内根据输入的先后，按照从左到右，再从上到下的顺序进行排列。也可以用 float: left 的定义直接将元素放置于左端；用 float: right 的定义将元素放置于右端。

2) 相对定位

相对定位(position: relative)属于普通流中的一种特殊情况。新元素的位置在普通流的

基础上再加上 top 和 left 坐标的偏移量。

例如图 4.2 中的第二个元素利用相对定位的结果，在普通流的基础上增加了(x,y)的偏移量(top:20px left:20px)。

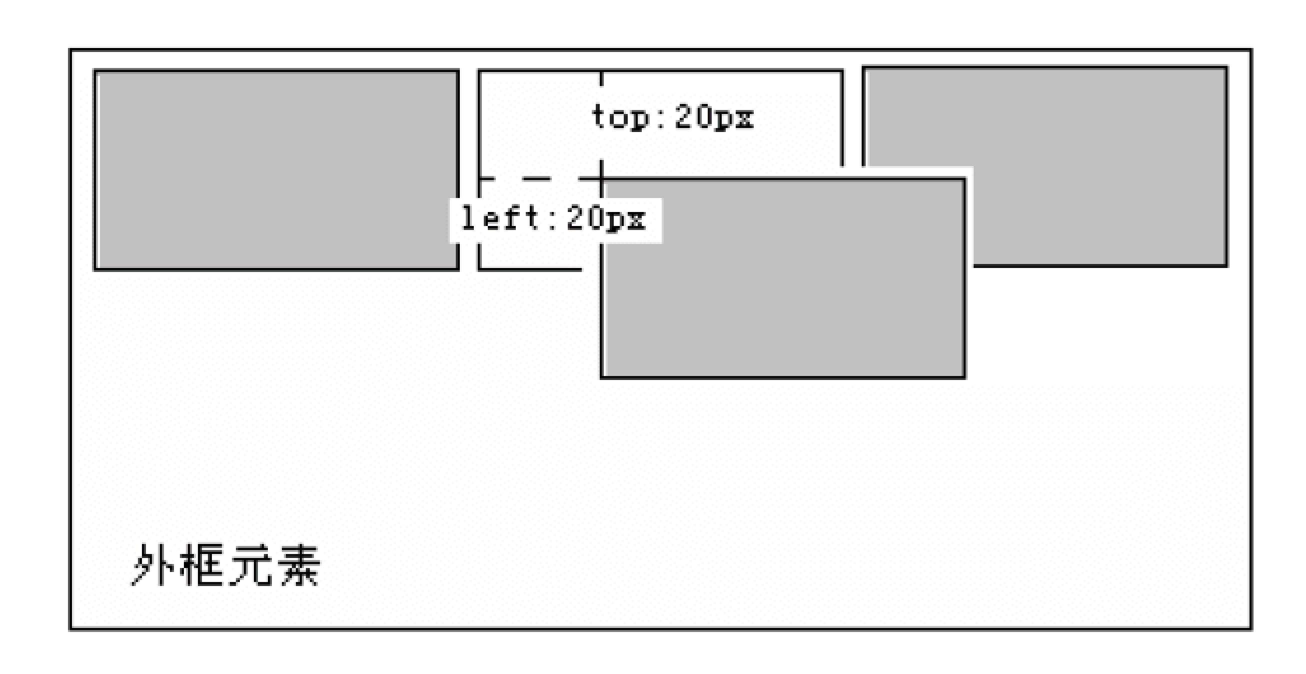


图 4.2　相对定位示例

此时 CSS 的定义如下：

```
#myBox {
        position:relative;
        left:20px;
        top:20px;
        }
```

使用相对定位可以更加灵活地调整元素的位置。但是不论元素如何移动，框架元素仍然保留了移动前由普通流占据的空间。

3)　绝对定位

绝对定位的方法比较简单，每个元素定位的 x、y 坐标都是以外框的左上角作为原点的偏移量。被绝对定位的对象从其他对象中分离出来，它脱离了文件流的影响。同时，它也不会影响其他对象的定位，它就像一张图层浮现在网页上面。

5. 常用的布局类型

网页中常用的布局有以下三种类型，每个类型都可以分为上、中、下三行。

- 上面一行通常为页眉(Header)：页眉中主要包括网页的标志、一些主菜单等。
- 中间一行通常为内容(Content)：中间是网页的主题内容，根据需要可能被划分为 1 列、2 列或 3 列，或其他几种不同的情况。
- 下面一行通常为页脚(Footer)：主要包括网页发布的日期、版权等相关事项。

几种常用的布局情况如图 4.3 所示。

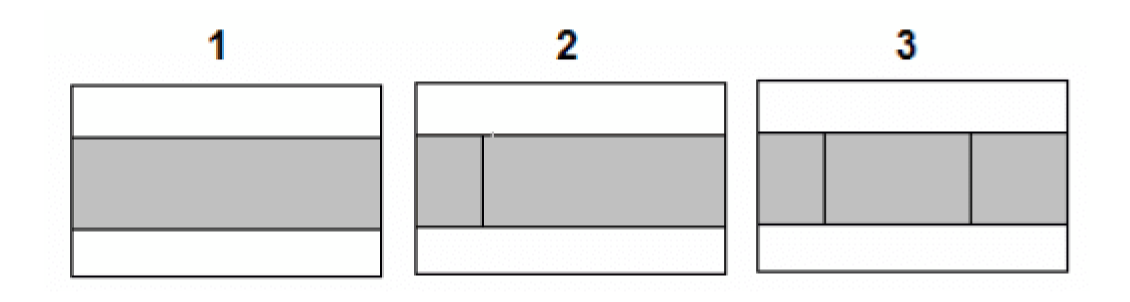


图 4.3　几种常用的布局

4.3.3 利用 ASP.NET 3.5 的工具进行布局

下面将通过一个典型的示例来讲解利用 ASP.NET 3.5 的工具进行布局的方法。本例是利用 CSS 进行三行、三列的布局(见图 4.4)。其步骤如下。

(1) 用 div 标记划分出页眉、内容、页脚三行，如图 4.4 所示。

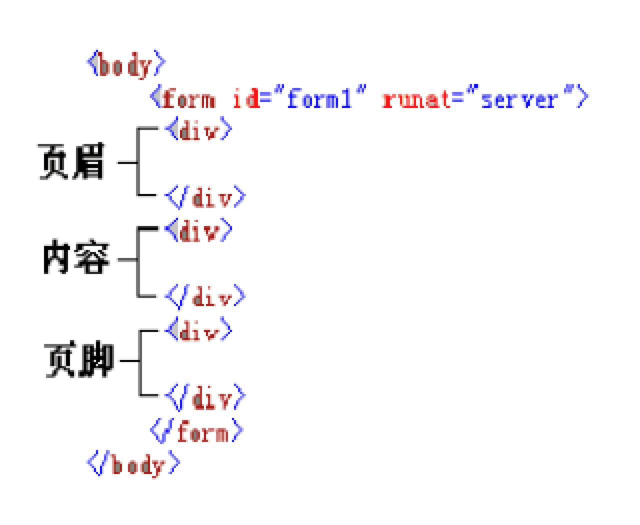


图 4.4 利用 CSS 布局三行

(2) 选择页眉的<div>标记后，转向【设计】视图。打开【管理样式】对话框，选择【新建 CSS】项，打开【新建样式】对话框，如图 4.5 所示。

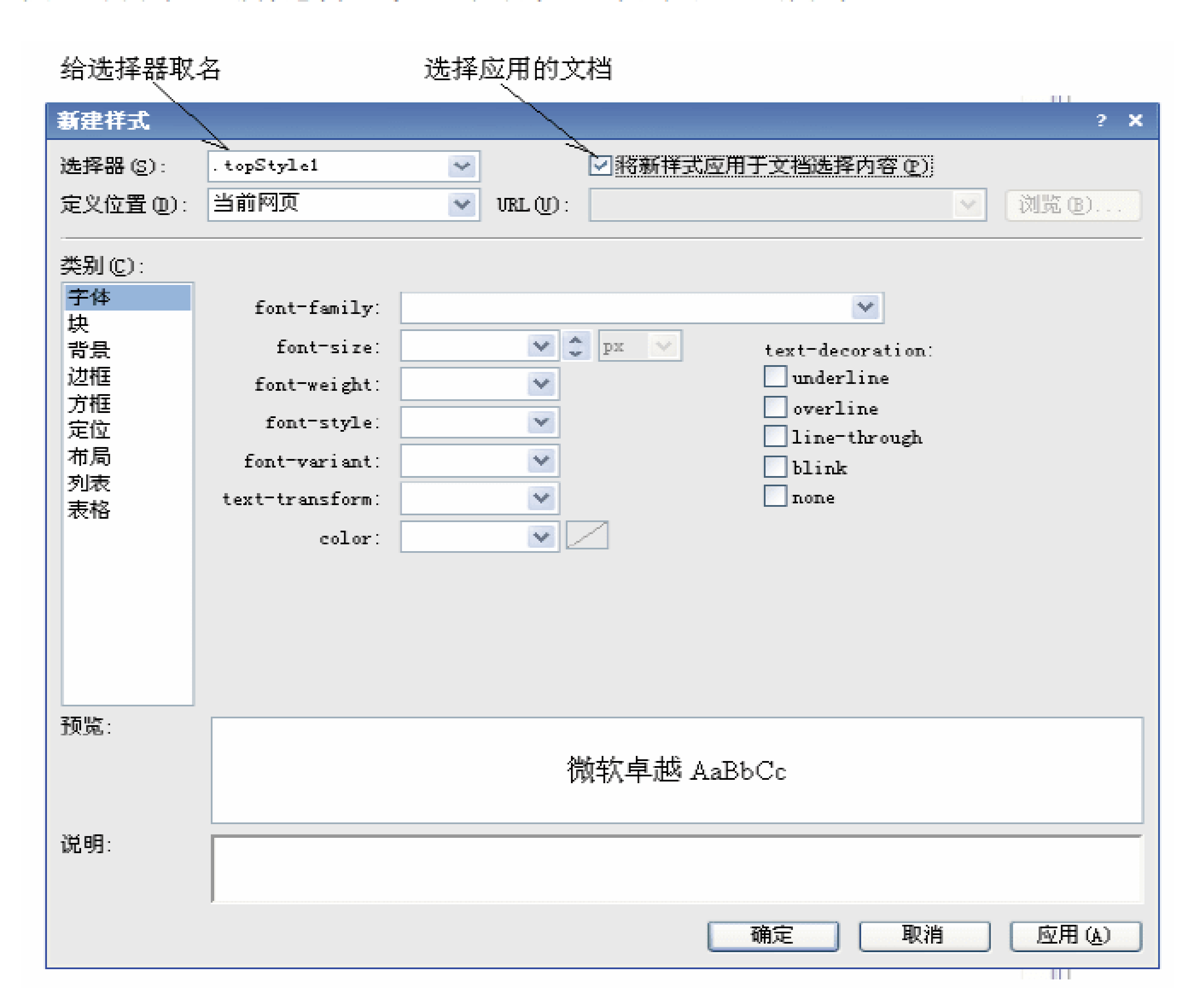


图 4.5 【新建样式】对话框

窗口的左边有九大项，右边是各大项的子项，可以在各种子项的下拉菜单中选择对应的值。首先在上方给选择器取名(这里使用类名.topStyle1)，并选择右边的选项，表明此选择器与前面选择的标记(div)连接。也可以在这里不选此复选框，以后通过 HTML 标记再进行连接。

(3) 选择左边的【定位】选项，打开相应的对话框，如图 4.6 所示。

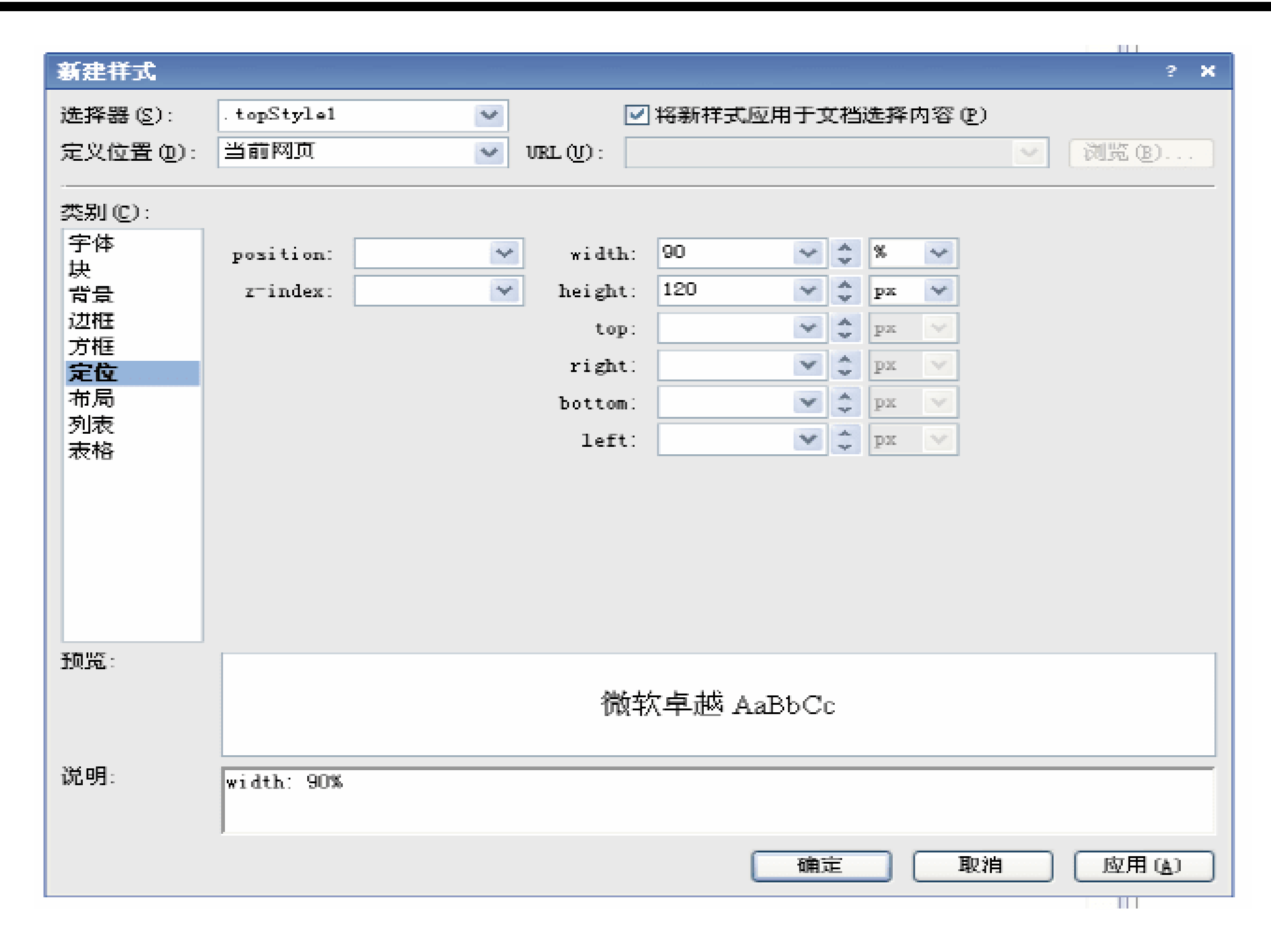


图 4.6　利用【定位】选项确定宽度(width)与高度(height)

在这个对话框中，将宽度(width)设为 90%，高度(height)设置为 120px。注意这里选择了不同的长度单位。高度在这里也可以不设，待回到【设计】界面后再用鼠标直接拖动来确定。

(4) 选择左边的【方框】项，打开相应的对话框。这里的方框就是前面讲述的框架模型，如图 4.7 所示。

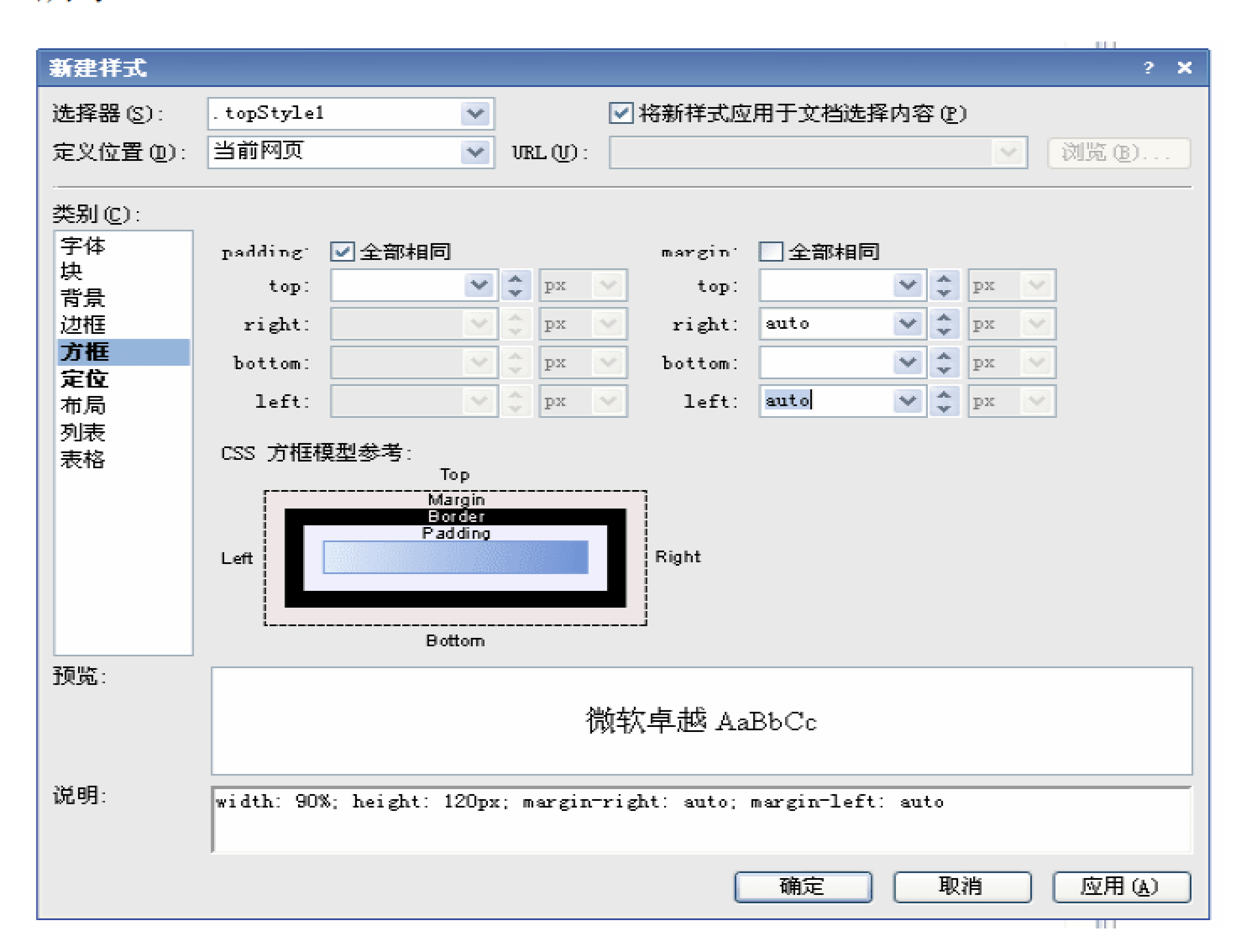


图 4.7　利用【方框】选项确定元素的框架模型

在这个对话框中可以设置框体的内填充(padding)和外空白边(margin)。默认情况下它们的四周都设置相同的值。如果想设置成不同的宽度时，先要取消选中【全部相同】复选框。在这里取消了 margin 项中【全部相同】的选择，并将其中的 right 和 left 属性均选用 auto。这是一项重要的技巧，这样设置的结果，可以保证元素在左右方向上居中。

(5) 为了显示设置的结果，可以为元素设置相应的边框。为此选择左边的【边框】选项，打开相应的对话框，如图 4.8 所示。

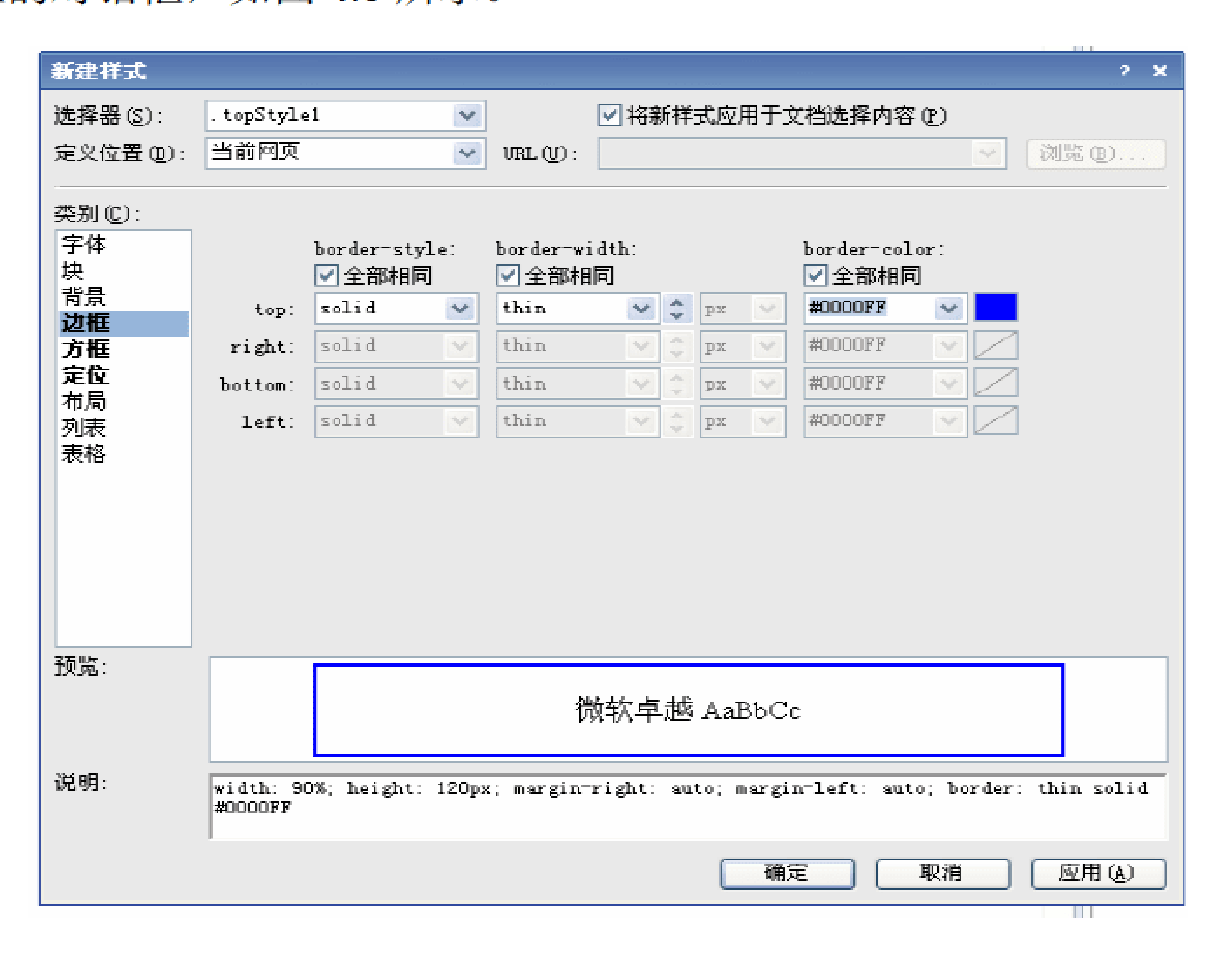


图 4.8 利用【边框】选项设定边框

将边框的样式(border-style)选为实线(solid)；将实线的宽度(border-width)设置为细线(thin)；将边框的颜色(border-color)设置为蓝色(#0000FF)。

设置内容层和页脚层的方法与前面设置页眉时大致相同，只是设置的具体数据稍有区别而已。设置的方法这里不再重复。设置的结果如图 4.9 所示。

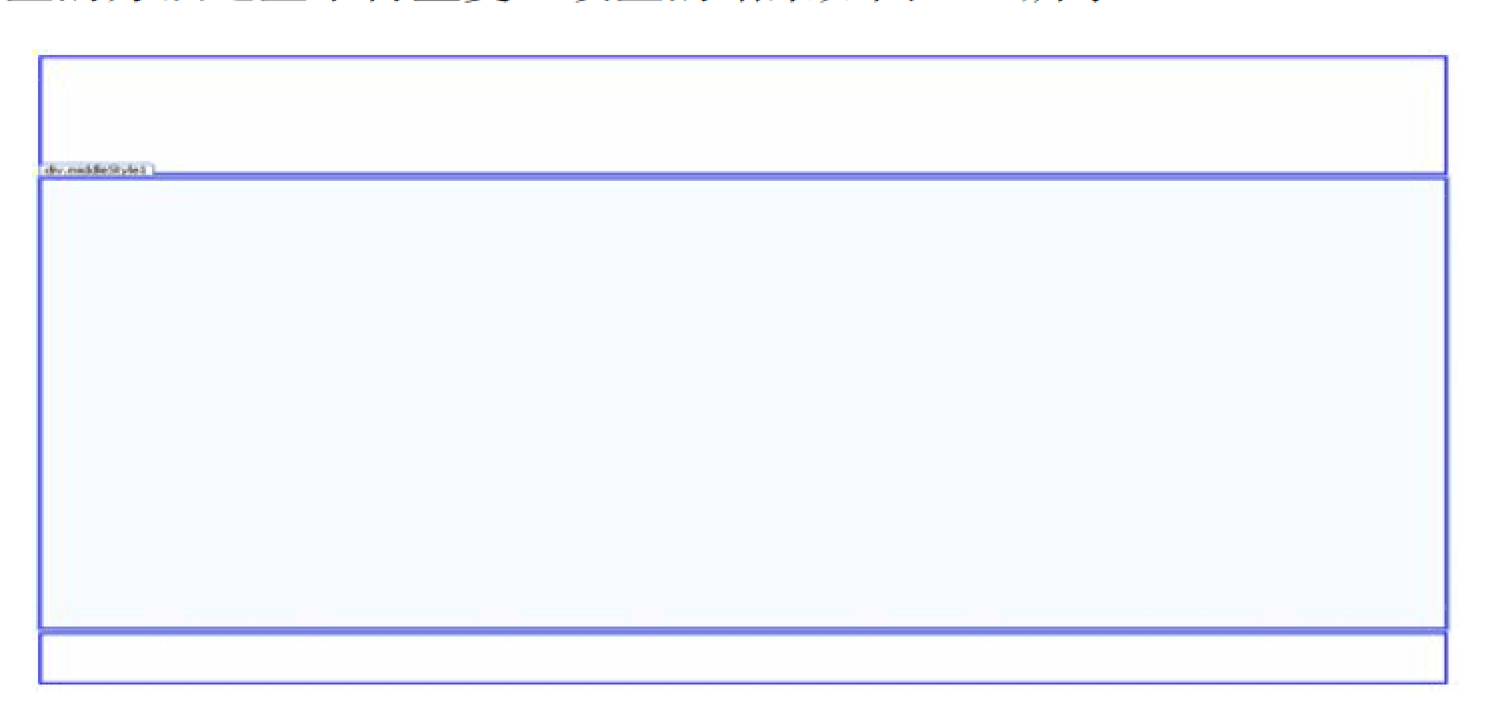


图 4.9 三行设置完后的显示界面

设置后再打开网页的【源】视图时，可以看出经过前面的设置，CSS 已经定义，各 div 标记已经通过属性中的类(Class)与定义连接。CSS 的定义如下。

```
<head>
  <style type="text/css">
       .topStyle1
       {
          width: 90%;
          height: 120px;
```

```
            margin-right: auto;
            margin-left: auto;
            border: thin solid #0000FF;
        }
        .middleStyle1
        {
            width: 90%;
            height: 466px;
            margin-right: auto;
            margin-left: auto;
            border: thin solid #0000FF;
            margin-top: 3px;
        }
        .bottomStyle1
        {
            width: 90%;
            height: 50px;
            margin-right: auto;
            margin-left: auto;
            border: thin solid #0000FF;
            margin-top: 3px;
        }

    </style>
</head>
<body>
    <form id="form1" runat="server">
    <div class="topStyle1">

    </div>
    <div class="middleStyle1">

    </div>
    <div class="bottomStyle1">

    </div>
    </form>
</body>
```

如果想修改 CSS 的定义，可以选用下列办法之一。

- 直接修改 CSS 定义中的代码。
- 在样式管理器中右击选择器名，在弹出的快捷菜单中选择【修改样式】命令，然后在弹出的对话框中改变设置。

(6) 在内容行中划分出三列。

为了将一行划分成多列并且将各列放到适当的位置，应注意框架排列的方式。比如分成两列时，一个靠左(float: left)，另一个靠右(float: right)即可。划分成三列时，一个靠左(float: left)，另一个靠右(float: right)，中间留出足够的空间，让中间列去自动填满。划分成 4 或 5 列时可采用 2+2 或 2+3 的办法，其他情况以此类推。

下面介绍划分三列的设计方法。 先将“内容”行划分成左、右、中间三列，如图 4.10 所示。

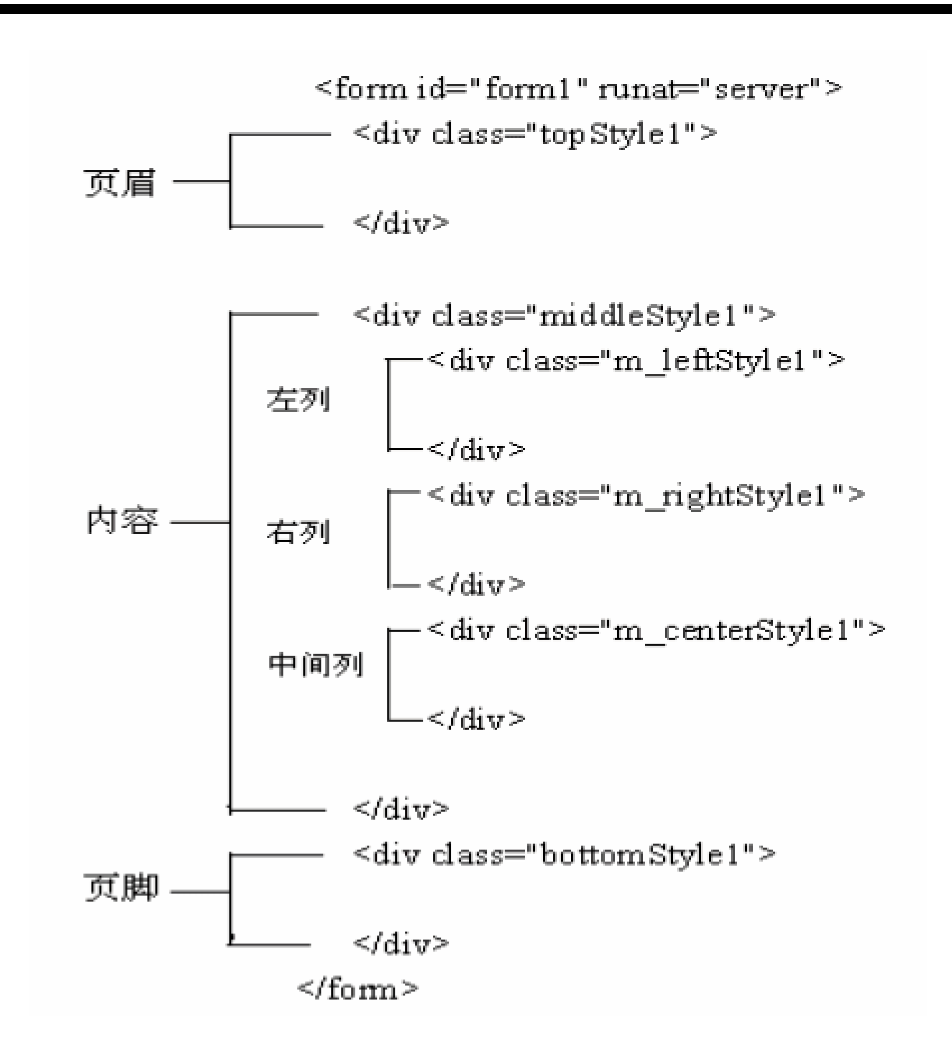


图 4.10　利用 CSS 给中间行划分成三列

利用 ASP.NET 3.5 布局的过程与前面相似，在定义左列时选用【布局】项后，在 float 下拉列表框中选择 left (靠左)，如图 4.11 所示。

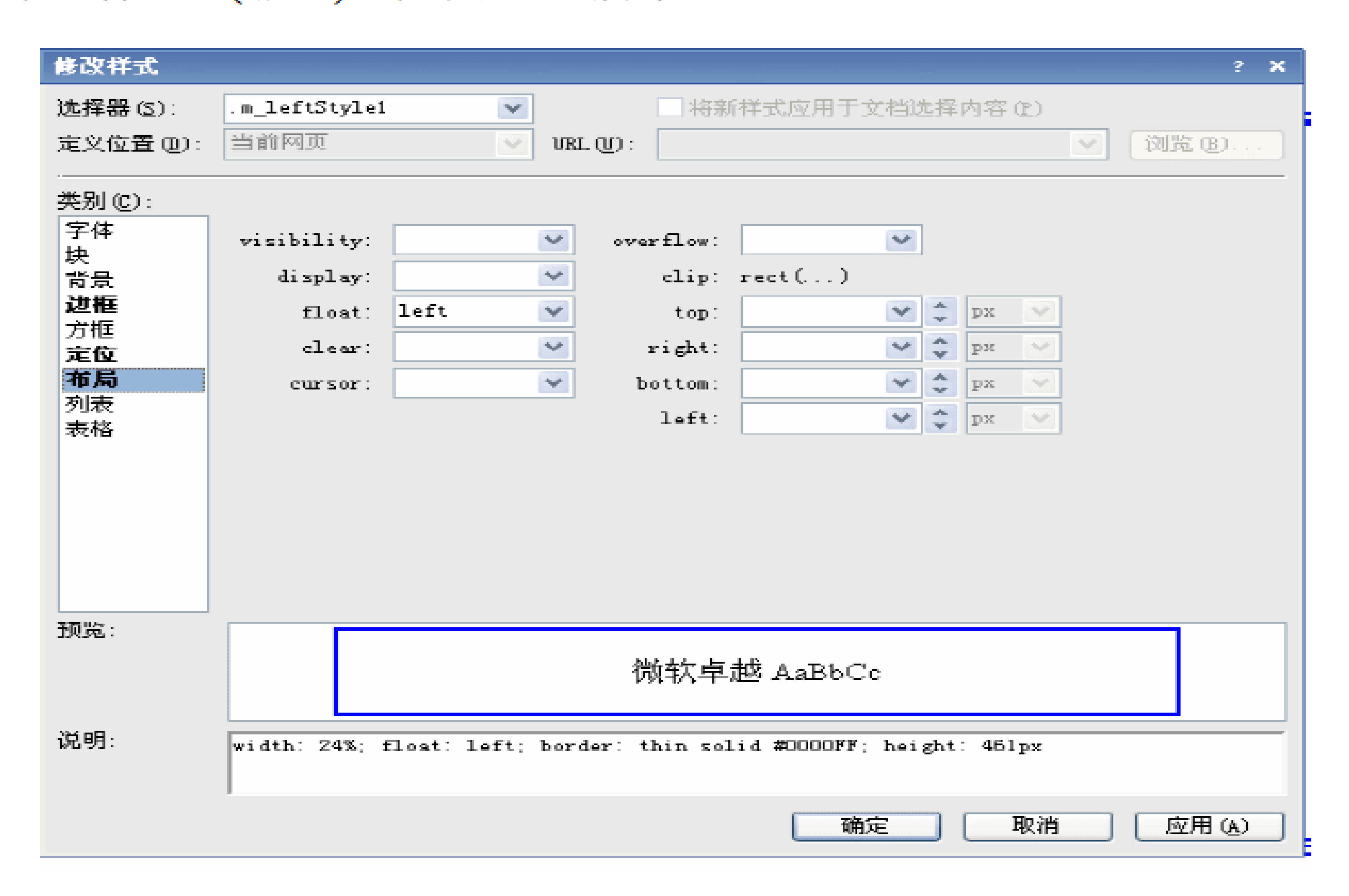


图 4.11　利用【布局】选项使元素靠左摆放

在定义右列时选用【布局】项后，在 float 下拉列表框中选择 right (靠右)。定义中间列时不定义宽度，也不定位，让它自动填满剩余空间。

注意代码的顺序。由于前面代码的顺序是先安排左列，再安排右列，最后才安排中间列(见图 4.10)，因此，在左、右两列的宽度已经确定的情况下，中间列不需要确定宽度就会按照框架模型自动填补剩余空间。如果代码的顺序有所改变时，中间列的宽度也应该确定具体值。

上述设置后的 CSS 代码如下。

```
.m_leftStyle1
```

```
    {
     width: 24%;
     float: left;
     border: thin solid #0000FF;
     height: 460px;
    }
  .m_rightStyle1
    {
     width: 20%;
     float: right;
     border: thin solid #0000FF;
     height: 460px;
    }
  .m_centerStyle1
    {
     border: thin solid #0000FF;
     height: 460px;
     }
```

4.3.4　对内容溢出的处理

由于有的网页内容始终处于动态变化之中，因此很可能出现元素内容超出其外框的情况，对于这种情况如果不作适当处理，内容将会冲破外框，从而打乱整个布局。

CSS 中的 overflow 是一个非常重要的属性，利用它可以非常方便地处理各种溢出情况。该属性允许对溢出采用 5 种不同的处理方式。

- hidden：将超出部分自动隐藏。
- visible：超出部分继续显示。
- scroll：在外框中生成上、下拖动条和左、右拖动条，允许利用拖动来显示超出部分。
- auto：当内容超出外框时，自动生成上、下或左、右拖动条。允许用拖动方法显示超出部分。
- inherit：继承父元素的设置。

为了防止溢出，在可能溢出的外框中，应该设置 CSS 的 overflow 属性。可以直接在 CSS 定义中编写代码。例如上面的“中间列”(选择器为.m_centerStyle1)中可能出现溢出，此时的 CSS 的定义如下。

```
.m_centerStyle1
          {
           border: thin solid #0000FF;
           height: 460px;
           overflow: auto;
           }
```

也可以利用前面的工具进行设置，方法是：先选择【布局】选项，然后在 overflow 下拉列表框中选择对溢出的处理方式。

4.4 小　结

CSS 是 HTML 的发展和补充，目前已经成为各类浏览器的共同标准。CSS 具有很强的表现力，利用它设计网页外观可以使网页变得更加美观，并为 DHTML(第 5 章讲述)奠定了基础。更为重要的是，它使网页内容与外观代码分离，使代码的结构变得更加清晰，有利于代码的重用。

CSS 的定义有两种形式：内联方式与链接方式。其中以链接方式为主，链接时各元素通过选择器与 CSS 定义相连。内联方式时将 CSS 的定义直接写在元素内部的属性中，这种方式是链接方式的必要补充。

框架模型以及它的排列方式是 CSS 布局的两个重要基础。框架的排列方式主要有三种：普通流、相对定位与绝对定位。三种方式各有不同的特点，分别适用于不同的场合。必须搞清它们的概念，并反复实践才能真正掌握。

4.5 习　题

1. 填空题

(1) CSS 全称为____________________，一般称为层叠样式表或级联样式表。简单地说，CCS 就是一组用来控制网页__________的属性。

(2) 在网页中引用外部的 CSS 文件时的语句是

```
<link rel=" _________" type= " _________" href="CSS 文件名.css" />
```

(3) 框架模型的长度单位主要有__________、_________、__________三种。

(4) 用 CSS 布局的方框模型中 margin 代表图 4.12 的______部分空间。

(5) 用 CSS 布局的方框模型中 padding 代表图 4.12 的______部分空间。

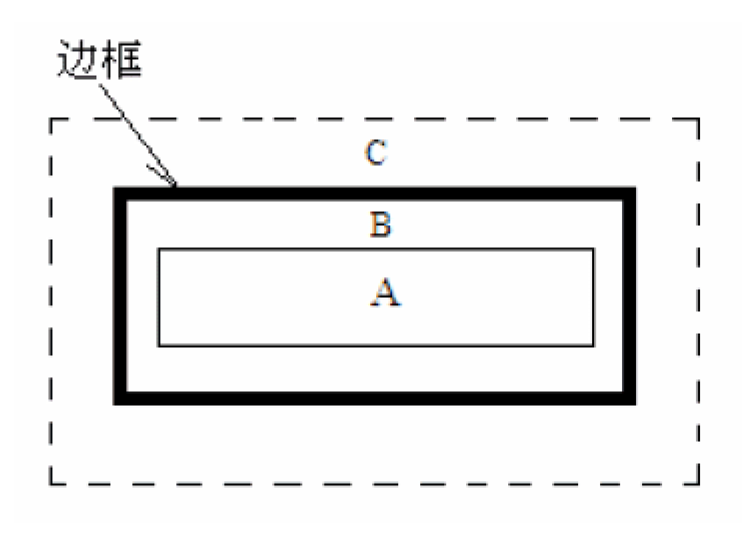


图 4.12　习题图

(6) 用 CSS 布局时，元素排列的方式主要有三种，它们是__________、__________、________。

2. 选择题

在 CSS 的 overflow 属性中，visible 代表_______，hidden 代表________，scroll 代

表______，auto 代表________。

A. 生成拖动条　　B. 显示

C. 超出范围时自动生成拖动条　　D. 隐藏

3. 判断题

(1) 为了实现内容与外观代码分离 CSS 应以链接方式为主。 (　　)

(2) CSS 的定义有了链接方式不再需要内联方式。 (　　)

(3) 利用 CSS 布局时相对定位与绝对定位方式只是坐标的原点不同。 (　　)

(4) 利用 CSS 布局时普通流与相对定位之间没有联系。 (　　)

(5) 利用 CSS 布局时普通流与绝对定位之间没有联系。 (　　)

4. 简答题

(1) CSS 有哪些特点?

(2) 使用 CSS 的方法有几种? 每种方法的思路和步骤是什么?

(3) 用 CSS 布局的方框模型中 margin 的作用是什么?

(4) 用 CSS 布局的方框模型中 padding 的作用是什么?

5. 操作题

利用 CSS 对网页进行布局。要求:

(1) 页面分三行，中间行分三列。

(2) 内容在左右方向上居中，并能随整个网页面积的改变自动升缩。

(3) 内容既包括文本又包括图片(图片应有边框和边框的内外间隙)。

(4) 能够自动处理内容溢出的问题。

第 5 章　动态 HTML 技术

动态 HTML 技术又称为 DHTML 技术，利用这个技术可以使下载后网页中的元素动起来，从而增强网页的生动性并增强网页的功能。

本章将要介绍的主要内容包括:

- 动态 HTML 的基本理论。
- JavaScript 语言。
- DHTML 的应用示例。

5.1　动态 HTML 的基本理论

5.1.1　DHTML 基本概念

1. DHTML 的引入

在早期的 Internet 体系中，网页运行的主要方式只是文本发布。服务器将文本、图像嵌在 HTML 标记中传送给浏览器，浏览器解读后按 HTML 标记的定义显示数据。在这里，服务器提供数据，浏览器显示数据。网页的内容和形式在服务器发出时已经确定，浏览器下载后不能再行改变，如果想改变网页上某些元素的表现形式，例如想改变某个元素的底色，也只有重载这张网页，以便启动服务器提供的服务。

这是一种静态关系，但这种静态关系对于初期的应用来说矛盾并不太大，因为初期使用网页的多数是一些科学工作者，他们中的大多数只是浏览网页中的内容，对网页的表现形式并不十分注重。随着网站应用领域的扩大，对网页的交互和动态方面的要求越来越高，静态网页已经远远不能满足客观的需要。

动态网页的出现，适应了形势发展的需要，是网页设计的重大突破。动态网页技术包括服务器端和浏览器端两个方面。

1)　服务器端的动态技术

为了实时响应客户要求，及时更换网页内容(例如更换数据库记录等)，产生了服务器端动态技术。从 CGI、ASP、JSP 一直到现在的 ASP.NET 和 Java/J2EE，服务器端动态技术有了很大的发展，这部分是后面的章节中将要重点介绍的内容。

2)　DHTML 技术

浏览器端的动态技术又称 DHTML，它是 Dynamic HTML(动态 HTML)的缩写。DHTML 技术近几年来发展也很快。它的设计思想是：浏览器从服务器端下载文档后，利用浏览器本身的资源，在不增加服务器负担和网上传输流量的前提下，使网页的某些元素“动”起来。

2. DHTML 是一项综合技术

DHTML 不是一种单一的技术，而是多项技术的综合。在浏览器对象模型(DOM)的基

础上，包括 HTML、CSS、事件、脚本等技术。其中 HTML、CSS 主要用于控制元素对象的属性。事件与脚本相结合主要用于控制元素对象的行为。通过这些技术可以改变网页元素的以下几个方面。

- 动态内容(Dynamic Content)：动态地增加、删除和修改文本、图形。例如，光标通过图片时，图片自动进行切换等。
- 动态样式(Dynamic Style)：动态改变文本、图像的样式。如改变它们的字体、颜色等。
- 动态定位(Absolute Positioning)：将事件、脚本、CSS 等技术相结合可以改变元素的位置。

除此之外，DHTML 还是一个开放型的系统，它将 ActiveX、动态 Firework、Flash、Java Applet 等一些成熟的多媒体技术集成在一起，从而把原来静态、呆板的网页变成一个丰富多彩的艺术品。

值得再次强调的是，通过 DHTML 技术使网页元素由静态转变为动态，是依靠浏览器本身的资源来完成的，它既没有增加服务器的负担，也没有加重网络上信息的传输容量。

5.1.2　DOM

1. DOM 概述

DOM(Document Object Model，文档对象模型)是 DHTML 的基础。

什么是 DOM？DOM 是 W3C 制定的标准，已为 Internet Explorer 4.0、Netscape 4.0 及以后各种版本的浏览器所接受。按照这个标准，浏览器端接收的各类网页元素，不仅仅只是一种显示格式，更是一个个“对象”。就是说，每个网页元素(对象)都有自己的属性和行为。通过对这些属性和行为的控制可以改变它们的状态和动作。

整个 DOM 是一种由对象组成的层次结构，就像一棵倒立的树(树根在上)，这棵树就称为文档对象模型(DOM)。各类浏览器的 DOM 结构稍有不同，Internet Explorer 4.0 的 DOM 结构如图 5.1 所示。

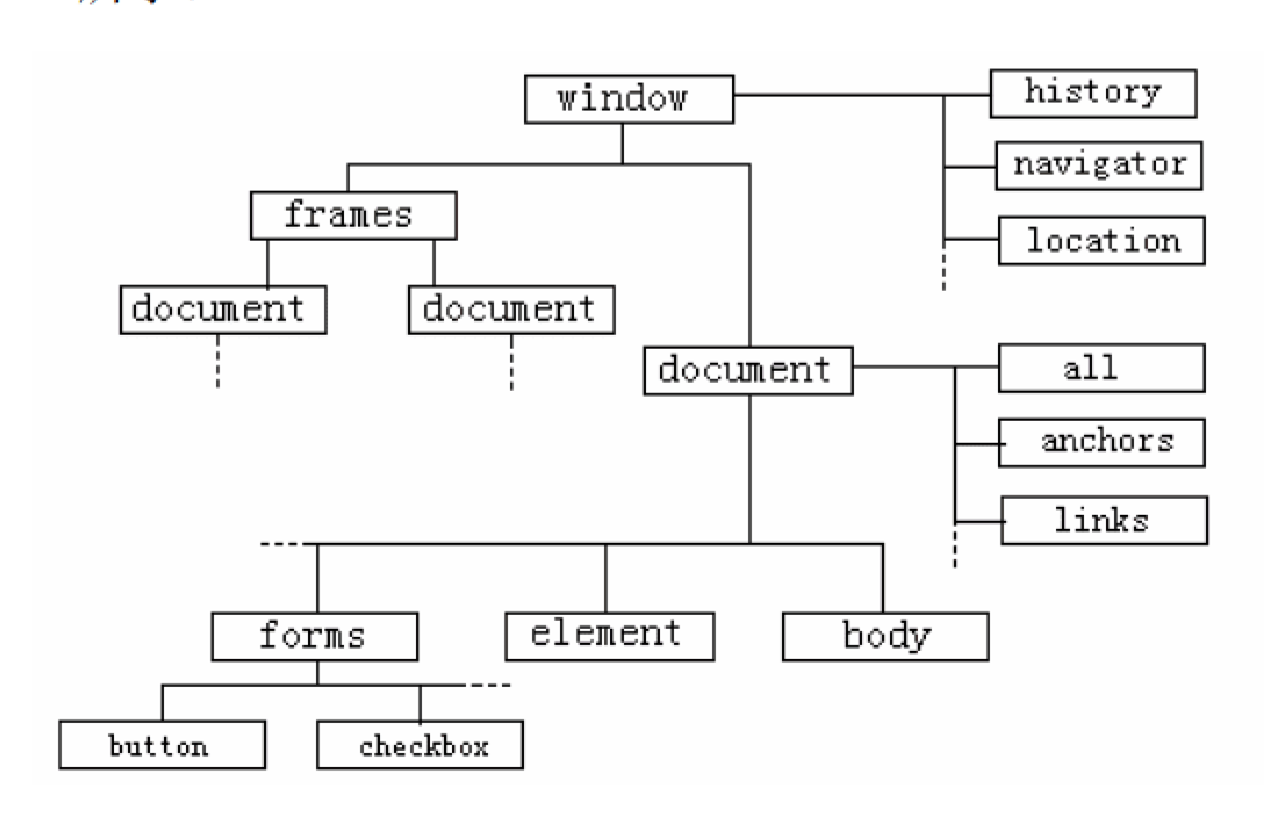


图 5.1　Internet Explorer 4.0 的 DOM 结构

在这个层次模型中，每个对象都是它父对象的属性。例如，window 对象是 document 对象的父对象，所以在引用 document 对象时使用 window.document。在这里，document 对象就相当于 window 对象的属性。

对于每一个网页，浏览器都会自动创建 window 对象、document 对象、location 对象、navigator 对象和 history 对象。基于这个层次结构，还可以创建其他对象。对于某个对象的属性，有时需要通过对象的完整路径来引用。

window 对象在层次中位于最高一层，具有唯一性，浏览器中的所有内容，包括页面及其他浏览器中设置的信息都存放在 window 对象或者它的子对象中。一般情况下，所有的脚本操作都是假定在当前窗口中进行的，所以调用 window 对象的方法时或者引用 window 的属性时，可以省略 window 对象的引用。例如，window.alert()可以简写为 alert()，window.document.write()可以简写为 document.write()。

2. window 对象的属性和方法

只要浏览器窗口打开，即使还没有加载任何页面，也会自动建立 window 对象。

1) window 对象的属性

(1) closed 和 opener 属性

通过 window 对象的 closed 属性可以判断一个窗口是否已经被关闭。在很多情况下，这种判断是必要的，因为如果一个窗口已被关闭，需要完成一些不同的操作。而如果一个窗口是通过 open()方法打开的，那么，在 opener 中存放的是所打开的它的父窗口，通过 opener 属性可以来操纵它的父窗口。当一个窗口打开另一个窗口后，子窗口只能通过 opener 属性和父窗口发生联系，父窗口只能通过 open()方法的返回值和子窗口发生联系，这两个相关窗口之间就这样实现“互操作”。

例如，通过 opener 属性可以获取父窗口的属性信息，下面的语句可以显示父窗口的名称。

```
alert(opener.name);
```

下面的语句可以判断一个窗口的父窗口是否已经被客户关闭。

```
if  (window.opener.closed)  {
 父窗口已经被关闭，进行相应的处理
 }
 else {
父窗口还没有被关闭，进行相应的处理
}
```

(2) defaultStatus 和 status 属性

defaultStatus 属性的值是在浏览器窗口下面的状态栏中默认的显示信息，status 属性是状态栏中当前显示的信息。

(3) document、history 和 location 属性

window 的 document 属性、history 属性、location 属性就是文档对象模型的 document 对象、history 对象、location 对象，两者是一致的。

2)　window 对象的方法

(1)　open()和 close()方法

使用 open()方法能够打开一个窗口，并且还能指定窗口的显示风格。open()方法将返回对窗口的引用，包含三个参数：页面地址、窗口名称、窗口风格。

在窗口风格中可以用 yes 或 no 来指定浏览器是否拥有工具栏(toolbar)、地址栏(location)、目录图标(directory)、状态栏(status)、菜单条(menubar)、滚动条(scrollbar)等。在窗口风格中还可以指定窗口的宽(width)和高(height)。

使用 close ()方法可以关闭一个窗口。

(2)　alert (字符串)、confirm (字符串)和 prompt (字符串，默认值)方法

在 JavaScript 编写的脚本中，常使用这三种方法和客户进行交互。其中：

alert()方法用于弹出警告框，在警告框中显示字符串。

confirm()方法用于弹出确认框，在确认框中显示字符串。例如：

```
if (confirm("Are you sure to submit ? ")) {
   执行表单的提交数据工作
}
```

prompt()方法用来弹出输入框，框内有默认值，输入框上面有提示用的字符串。例如：

```
var n_name = prompt("Type in your name please: ", "wang");
```

语句 Type in your name please 是输入框上面的提示，wang 是默认值。当在输入框中输入新值并且单击 OK 按钮时，将把输入的值赋给变量 n_name。

3. document 对象

document 对象代表当前整个网页，在 document 对象中存储着当前页面的一些信息，包括页面的前景颜色和背景颜色，也包括页面中的表单、锚标、图像等对象。通过 document 对象，还可以向页面中动态添加文本以及各种标记。

1)　document 对象的属性

document 对象的属性非常丰富，下面分类简要介绍一下。

(1)　bgColor、fgColor、linkColor、alinkColor 和 vlinkColor 属性

属性都是用于颜色设置。其中 bgColor 是网页背景的颜色，fgColor 是网页内文本(前景)的颜色，linkColor 是超链接字符串的颜色，alinkColor 是单击超链接时的颜色，vlinkColor 是已经访问过的超链接的颜色。

这里的颜色都要以 0xrrggbb 形式表示。其中 0x 代表十六进制，rr 代表红色的深浅程度，gg 代表绿色的深浅程度，bb 代表蓝色的深浅程度。其范围为 00～ff。其他颜色都由这三种颜色组合而成。例如设置红色时用 0xff0000 表示。

(2)　title 属性

title 是页面标题，也就是在 HTML 中<TITLE>标记中设置的标题，可以使用以下代码设置它。

```
document.title = "Shopbag";
```

(3) anchors、applets、forms、images 和 frames 属性

属性是用来存放对象的数组。网页中所有超链接都存放在 anchors[ ]数组中；所有 Java Applet 都存放在 applets[ ]数组中；所有表单都存放在 forms[ ]数组中；所有图像都存放在 images[ ]数组中；所有框架都存放在 frames[ ]数组中。

利用数组可以引用数组内的对象。例如用 document.forms[0]来引用第一个表单对象。

2) document 对象的方法

(1) write ()和 writeln ()方法

write ()方法用于向网页内写入文本或者标记。writeln ()方法的作用与 write()相同，不同之处是后面包括一个换行符(回车)。不过这个换行符 HTML 并不认识，要想在 HTML 中换行，还需要用 write ()或 writeln ()方法写入
这个 HTML 中的换行标记。

(2) open()和 close()方法

open()方法用于打开一个新文档，close()方法用来关闭当前文档。通常情况下要求使用 write()或 writeln()方法前先用 open()方法打开文档，写完以后用 close()方法关闭，以保护文档的安全。

4. location 对象

在 JavaScript 中，当前浏览器访问页面的 URL 地址存放在 location 对象中，使用 location 对象，可以对这个 URL 进行分析，并将浏览器引导到指定地址。

1) location 对象的属性

假定有一个这样的 URL 地址：

```
http://www.myset.com.cn:80/welcom/index.htm#section2?id=3
```

其中协议名称为 http，主机名为 www.myset.com.cn，端口号为 80，页面地址为 welcom/index.htm，在这个网页内有一个锚标，名称为 section2。使用 location 对象能够分析这个 URL 地址的各个部分。

(1) protocol 属性

protocol 属性指明了通信的协议。在网页中通常都是采用 http，除此之外，还有 ftp 和 gopher 协议。

(2) host 属性

host 属性指明了页面所在 Web 服务器的主机名称，可以是域名，也可以是 IP 地址。

(3) port 属性

port 属性指明了服务器通信的端口号，一般在 URL 中不注明，默认情况下这个端口号是 80。

(4) pathname 属性

pathname 属性指明了页面在服务器上的路径以及页面文件的名称。

(5) hash 属性

如果页面需要跳转，在 URL 地址中将包括锚标，以便跳转到指定的部分。此时可以利用 hash 属性获得页面跳转的锚标的信息。

(6) search 属性

在 URL 后的问号后面常有一些信息(如上例中的?id=3)。这个信息是提交到服务器上

进行搜索的信息。

(7)　hostname 属性

hostname 属性将 host 属性与 port 属性结合在一起，既包括主机名，又包括主机端口号，主机名与端口之间用冒号(:)分隔。

(8)　href 属性

href 属性提供整个 URL 地址，这个属性将上面几个属性信息结合在一起。

2)　location 对象的方法

使用 location 对象的方法能够对页面进行刷新，或者将页面引导到另外一个 URL 地址。

(1)　assign(URL 地址)方法

利用 assign()方法可以将页面引导到另一个 URL 地址。例如：

```
location.assign("http://www.myset2_com.cn/index.htm")
```

将页面引导到 www.myset2_com.cn/index.htm 中去。

(2)　reload()方法

reload()方法用于对网页全面刷新。例如定义一个按钮，其功能就是对网页进行刷新。

```
<INPUT TYPE="button" VALUE="RELOAD" OnClick="location.reload()">
```

(3)　replace(URL 地址)方法

replace()方法可以用新的 URL 地址取代当前页面，它与 assign()方法的不同之处在于，assign()是将页面导航到另外一个页面，单击【后退】按钮时还能返回原来的页面，而使用 replace()方法则是用另一个页面取代当前页面，不能用【后退】按钮返回。

5.2　JavaScript 语言

5.2.1　JavaScript 语言简介

JavaScript 是由 Netscape 公司开发的一种解释型语言，作为一种脚本语言可以直接嵌入到 HTML 页面中，和 HTML 紧密地结合在一起。JavaScript 既可在浏览器又可在服务器端解释执行，而当前的主要浏览器 Netscape Navigator 完全支持 JavaScript 语言。Internet Explorer 也支持一种与 JavaScript 极为近似的 JScript 语言。因此 JavaScript 已经成为各类浏览器的通用语言。

JavaScript 是一种基于面向对象和事件驱动(Event Driver)的跨平台的脚本语言，现在 ASP.NET 已经将它改造成 JavaScript.NET，成为完全面向对象的语言。但是为了能够在各种浏览器中通用，JavaScript.NET 与原来的 JavaScript 完全兼容。

5.2.2　JavaScript 的基本用法

1. JavaScript 的基本语法

JavaScript 的语法与 C 或 C++语言近似。其具有如下特点。

- 命令、函数名、变量名都区分大小写。
- 每条语句后用分号结尾。但是 JavaScript 对这一点不像 C 语言那样严格，用 JavaScript 编写脚本时，语句后可以加分号也可以不加分号。建议加上分号，以养成良好的编程习惯。
- 每个过程都是一个函数。
- 语句块(多条相关的语句)用大括号界定。
- 单行注释前用双斜杆(//)表示，多行注释用/*…*/界定。

和 C 语言相比，最大的不同点是，由于 JavaScript 是一个比较松散的语言，变量使用前不一定要进行声明，即使声明时也不一定指定数据类型，运行中也不进行强制性的类型检查。这样的约定既有好处也有缺点。好处是书写代码简单，可以节省声明的时间，缺点是一旦发生了错误，不容易找到问题所在。好在 JavaScript 通常只用于编写小型程序(脚本)，因此这个缺点带来的后果并不严重。只是在处理数值与字符串表达式时，应记住一条原则：数值加数值结果仍然是数值，但数值与字符串相加时，数值将自动转换为字符串，然后与其他字符串拼接起来。

2. JavaScript 的数据类型

JavaScript 的变量区分大小写，就是说 name、NAME 和 Name 是三个不同的变量。变量名可用字母或下划线(_)开头，但不能用数字 0～9 作为变量名的开头。变量名的第二个以及以后的字符可以用数字、字符或下划线。

JavaScript 的数据类型有以下 4 种。

(1) 整型数 (Integer)通过三种形式来表示。

- 十进制(decimal)：dd…d 其中 d 为 0～9 的整数，但不能用 0 开头。
- 十六进制(hexdecimal)：0xhh...h，其中 h 为 0～9、A、B、C、D、E、F 的十六进制数。
- 八进制数(octal)：ooo…o，其中 o 为 0～7 的整数。

(2) 浮点数(Float)有两种表示方法。

- dddd.ddd：小数点表示法。小数点左侧为整数部分，右边为小数部分。
- dddd.dddE+ddd 或者 dddd.dddE-ddd：科学表示法。把一个数用 E(或 e)分隔开，左侧为真数，右侧为指数(以 10 为底)。

(3) 布尔值(Boolean)：只有两种值，true 和 false。

(4) 字符串(String)：用双(单)引号括起来的 0～n 个字符。

① 特殊字符：用一些特殊字符来表示某些控制码，如表 5.1 所示。

表 5.1 特殊字符及其含义

特殊字符	含　义	特殊字符	含　义
"\b"	退格(backspace)	"\r"	回车(carriage return)
"\f"	换页(form feed)	"\t"	跳过 8 个空格(tab character)
"\n"	换行(new line character)		

② 空字符：空字符 Null 是一个特别的值，代表空值(Null Value)。

3. JavaScript 的运算符

JavaScript 的运算符有以下 9 种。

(1) 基本运算符：加(+)、减(−)、乘(*)、除(/)。

(2) 余数运算符：%。例如，18 % 4，结果是 2。

(3) 递增(递减)运算符：++x、 x++、−−y、 y−−等。

++或−−代表变量本身“加 1”或“减 1”的操作。这两个符号可以放在变量名的前面，也可以放在变量名的后面，由于两种方法执行的顺序不同，结果有时是不一样的。

(4) 位逻辑运算符：位逻辑运算符有三种，如表 5.2 所示。

表 5.2　位逻辑运算符

运 算 符	用　法	说　明
&	X & Y	将 X 与 Y 对应的位进行与(AND)运算
|	X | Y	将 X 与 Y 对应的位进行或(OR)运算
^	X ^ Y	将 X 与 Y 对应的进行位异 (XOR)运算

(5) 移位运算符：移位运算符(<<、>>、>>>)的用法如表 5.3 所示。

表 5.3　移位运算符

运 算 符	用　法	说　明
<<	X<<Y	将 X 左移 Y 位，右侧补 0
>>	X>>Y	将 X 右移 Y 位，左侧用原来的首位补充
>>>	X>>>Y	将 X 右移 Y 位，左侧补 0

(6) 逻辑运算符：逻辑运算符有三种，即&&、\|\| 和！。它们的用法如表 5.4 所示。

表 5.4　逻辑运算符

运 算 符	用　法	说　明
&&	X && Y	X 与 Y 进行 AND 逻辑运算
||	X || Y	X 与 Y 进行 OR 逻辑运算
!	!X	对 X 进行 NOT 逻辑运算

逻辑与(AND)的真值表如表 5.5 所示。

表 5.5　逻辑与的真值表

X	Y	X&&Y
True	True	True
True	Flase	Flase
Flase	True	Flase
Flase	Flase	Flase

逻辑或(OR)的真值表如表 5.6 所示。

表 5.6 逻辑或的真值表

X	Y	X||Y
True	True	True
True	Flase	True
Flase	True	True
Flase	Flase	Flase

逻辑否(NOT)的真值表如表 5.7 所示。

表 5.7 逻辑否的真值表

X	!=x
True	Flase
Flase	True

(7) 比较运算符包括等于(=)、大于(>)、大于等于(>=)、小于(<)、小于等于(<=)、不等于(!=) 6 种，其用法如表 5.8 所示。

表 5.8 比较运算符

运 算 符	用 法	说 明
=	X==Y	X 等于 Y 时为 True，否则为 Flase
>	X>Y	X 大于 Y 时为 True，否则为 Flase
>=	X>=Y	X 大于或等于 Y 时为 True，否则为 Flase
<	X<Y	X 小于 Y 时为 True，否则为 Flase
<=	X<=Y	X 小于或等于 Y 时为 True，否则为 Flase
!=	X!=Y	X 不等于 Y 时为 True，否则为 Flase

(8) 字符串运算符：字符串运算符中“+”和“+=”可作为字符串的拼接符号。

(9) 其他操作符。

其他操作常用的有 4 种。

① 条件运算符。

```
结果 = (条件) ? value1 : value2;
// 假定条件为真时结果为 value1，否则结果为 value2
```

② new 操作符：new 操作符可以为客户自定义对象类型或者为 JavaScript 的内置对象创建一个实例。例如，可以用 new 操作符创建一个日期类型的对象。

```
var d_Date = new Date();
```

③ delete 操作符：delete 操作符可以删除对象、对象的属性或者数组中的元素。

④ this 操作符：JavaScript 利用 this 操作符来引用当前的对象。

4. 运算符的优先顺序

运算符的优先顺序从高到低排列如下。

(1)　括号()
(2)　求补、增量、减量(~、++、--)
(3)　乘法、除法、余数运算(*、/、%)
(4)　加法、减法(+、-)
(5)　移位运算(<<、>>、>>>)
(6)　比较运算(<、<=、>、>=)
(7)　相等比较(==、!=)
(8)　位逻辑的与(AND)运算(&)
(9)　位逻辑的异或(XOR)运算(^)
(10) 位逻辑的或(OR)运算(|)
(11) 逻辑与(AND)运算(&&)
(12) 逻辑或(OR)运算(||)
(13) 条件运算(?:)
(14) 赋值及复合运算(+=、-=、*=、/=、%=、<<=、>>=、>>>=、&=、^=、!=)
(15) 逗号运算(,)

5. JavaScript 的流程控制

1)　条件语句

(1)　条件语句的格式之一：if 语句

```
if (条件表达式)
{条件为真时执行的语句}
[else
{条件为假时执行的语句}]
```

(2)　条件语句的格式之二：switch 语句

```
 switch (表达式){
  case 标记1:
   代码块1;
   break;
  case 标记2:
  代码块2;
  break;
 …
[default:
   默认代码块;]
}
```

上述语句的执行过程是，先计算出表达式的结果，然后将结果与标记 1、标记 2…逐个进行比较，找到第一个能够匹配的标记，就执行该标记下的代码块。如果所有的标记均不能匹配时执行默认代码块。每个代码块后面都有一个 break 语句，它的作用是执行完本代码块后跳过后面的选择。如果没有这一句，执行完本代码块以后还要和后面的标记一一

比较，从而降低了程序的执行效率。另外程序中的 default 以及后面的默认代码块是可选的部分。如果这一部分没有写，而又找不到匹配的标记时，这段程序将什么事也不做。

当情况比较复杂时，用 switch 语句显得思路更加清晰一些。

2) 循环语句

JavaScript 支持两种循环结构：一种是 while；另一种是 for。另外 break 和 continue 语句也可以用于循环语句中。

while 循环语句的格式如下。

```
初始表达式
while (循环条件){
  代码块
  递增表达式
  }
```

do…while 循环语句的格式如下。

```
do {
代码块
} while (循环条件)
```

do…while 语句在执行顺序上与 while 语句不同。在 while 语句中是先判断再执行，而 do…while 语句却是先执行再判断。因此如果开始循环时条件就不成立，do…while 语句中的代码块还得执行一次；而 while 语句中的代码将一次也不执行。

for 循环语句的格式如下。

```
for ([循环变量赋初值]; [循环条件]; [循环变量增量])
 {代码块}
```

执行 for 循环有以下 5 个步骤。

(1) 对循环控制变量赋初值。

(2) 判断循环条件是否为真，若为真时进入循环，否则退出循环。

(3) 进入循环后执行代码块。

(4) 执行控制变量增量表达式。

(5) 返回第 2 步以判断循环条件是否为真。

在 for 循环语句中，[循环变量赋初值]、[循环条件]、[循环变量增量]都是可以选择的项目，因而都可以省略。按照语法规定，不论省略哪项，原有的分号不能省略。被省略语句的功能，应该用其他语句完成。

在循环代码块中使用 break 和 continue 语句的作用是：break 语句用于跳出循环；continue 语句用于跳过 continue 后面的语句，进入新一轮循环。

6. JavaScript 的函数

1) JavaScript 的全局函数

JavaScript 中定义了一些全局函数，这些函数不与任何具体对象发生联系，因此称全局函数。全局函数在脚本的编写中很有用处。

(1)　eval(字符串)

eval()函数的功能就是执行括号内的字符串，如果字符串代表一个表达式，那么函数将对这个表达式进行运算；如果字符串代表一条或多条语句，那么函数就执行这些语句。

(2)　parseInt(字符串)和 parseFloat(字符串)

parseInt()和 parseFloat()函数的作用是将括号内的字符串转变为整数和浮点数(小数)。网页中使用表单时，从文本框中取出来的值，都属于字符串类型，若想运算需先将它们转换成整数或浮点数类型，此时可利用这两个函数完成类型转换的工作。

(3)　splite()分离的字符串

```
var listArray = stringList.split(",");
```

函数的作用是将字符串(stringList)以逗号作为分隔符，分解成字符数组(listArray)，如 listArray[0]、listArray[1]等。

(4)　isNaN(表达式)

isNaN() 函数用于判断表达式是否为数值型，在使用 parseInt(字符串)和 parseFloat(字符串)函数时，如果字符串不代表一个数值，此时两个函数都会返回 NaN(NaN 的意思是 Not a Number，指出不是一个数字)。通常，在使用上述函数后，还需要用 isNaN 函数进行判断，看字符串是否数值，如果不是，需做其他处理。

(5)　Number(对象)和 String(对象)

Number()和 String()的参数是对象。Number()将对象转换为数值，String()将对象转换为字符串。与 parseInt 和 parseFloat 函数一样，如果转换失败，将返回 NaN。

(6)　escape(字符串)和 unescape(字符串)

escape()和 unescape()函数是一对功能相反的函数，对应于对字符串进行编码和解码的工作。浏览器对 URL 代码采用 ISO-Latin-1 字符串进行编码，此时的空格都用%，汉字都用一种很难识别的代码取代。可以采用 unescape()函数进行解码，还原成原来的样式。相反，也可以用 escape()函数编码，变成 URL 需要的格式。

2)　自定义 JavaScript 函数

在 JavaScript 中，函数是一种能够完成一定功能的代码块，函数可以在脚本中被事件或其他语句调用，函数执行完以后可以返回值，也可以不返回值。不返回值的函数就相当于其他语言中的过程。

(1)　函数定义

函数定义的格式是：

```
function 函数名(参数列表) {
    代码块
}
```

参数列表是可选部分，如果没有参数时使用一个空括号。如果函数需要返回一个值给调用函数时，代码块中应该包括 return 语句。函数允许嵌套，即在一个函数中还允许调用其他函数。

(2)　函数的调用

函数只有在被调用时才执行。有两种形式的函数调用语句。

- 函数名(参数列表)

- 变量名= 函数名(参数列表)

其中第一种形式适合于不返回参数的调用，第二种形式适合于返回参数的调用。

当脚本的代码很长时，应该根据功能将代码块分别写入几个函数中，使得每个函数的功能尽量单一，以增加程序的可读性，同时也便于脚本的编写和调试。如果在多个网页中需要使用同一函数时，可以将该函数单独提取出来写成单独的文件。然后在各个网页前面用 Include 语句将它和网页组合到一起。

7. JavaScript 语言中的对象

1) 数组对象

JavaScript 中没有数组这个数据类型，但在编写脚本时，数组的强大功能又是不可替代的。在开发脚本时可以使用数组对象来完成数组的功能。JavaScript 的数组对象除了具备其他语言中数组的功能以外，还有许多其他语言不具备的数组方法。

(1) 建立数组

建立数组有两种方法。第一种方法是，建立数组的同时给数组元素赋值。第二种方法是，建立数组时定义数组的长度，以后再给数组元素赋值。

JavaScript 对于语法的要求并不严格，允许不事先指定数组长度，而在以后的脚本语句中自动确定数组的长度。

(2) 访问数组元素

在 JavaScript 中是通过数组名及其下标来访问数组元素的。值得注意的是，数组下标从零开始。

(3) 数组对象的属性与方法

数组对象有一个属性，就是数组的长度。数组的方法有很多，下面介绍其中的两个。

第一个为 sort()方法。利用这个方法能够将数组元素进行排序。

第二个为 reverse()方法。利用这个方法可以将数组内的元素顺序颠倒过来，第一个元素与最后一个元素交换，第二个元素与倒数第二个元素交换，以此类推。

2) 字符串对象

在 JavaScript 中，有字符串类型的数据，同时也有字符串对象，它们的定义方法是不同的。当 st 是一个字符串类型的变量时，定义为 st="hello"；当 st 是一个字符串对象时，定义为 var st = new String("hello")。

字符串类型变量和字符串对象在使用时并没有很大区别，字符串变量也可以使用字符串对象的属性和方法。

字符串对象有一个属性 length，代表字符串的长度。字符串还有些常用的方法，现在介绍下面 4 种。

(1) toUpperCase()与 toLowerCase()方法

字符串的 toUpperCase()方法将使字符串的字母全部转换为大写；字符串的 toLowerCase()方法将使字符串的字母全部转换为小写。由于 JavaScript 是区分字母大小写的，所以有时需将字符串转变为同一类型(大写或小写)后再进行比较。

(2) indexOf(子字符串)方法

indexOf(子字符串)方法的作用是在母字符串中确定子字符串的位置。如果在母字符串

中包含着子字符串，将返回子字符串第一个字符在母字符串中的下标。如果母字符串中不包含子字符串时将返回-1。例如，母字符串为 s = "How are you! "，则 s.indexOf("are")将返回 4。注意字符串的下标从 0 开始。

(3)　charAt(位置)方法

charAt()方法中有一个位置参数。它的作用是返回字符串中该位置的字符。例如上面的 s 字符串，用 s.charAt(4)将返回字符 a。

(4)　substring(位置 1，位置 2)方法

substring(位置 1，位置 2)方法的作用是返回字符串中从“位置 1”到“位置 2”之间的字符串。返回的字符串中不包括位置 2 的字符。例如 s = "How are you!"，s.substring(8,10)将只返回 yo 两个字符。

3)　日期对象

JavaScript 中，经常要对日期进行处理，但 JavaScript 中没有日期类型，所以就会常用到日期对象。在 JavaScript 中，日期对象中所存储的不仅仅是日期数据，还包括时间数据。在这个对象中，实际存储的是相对于 1970 年 1 月 1 日 0 时 0 分 0 秒的毫秒数，如果对象代表的时间是在 1970 年以前，那么它存储的是负数，对于 1970 年以后的时间，日期对象存储的是正数。

(1)　创建日期对象

创建日期对象要用到 new 操作符。创建的一般格式如下：

```
var 对象名称= new Date(参数)
```

①　创建当前的日期和时间的对象

当没有参数时，可以使用日期对象获得当前的日期和时间。例如：

```
var d_Date = new Date();
```

②　创建指定日期的对象

在创建日期对象时，可以指定其中的年、月、日。例如：

```
var d_Date = new Date(2001,9,1);
```

值得注意的是，这里创建的对象并不是 2001 年 9 月 1 日，而是 2001 年 10 月 1 日，因为系统使用 0 到 11 分别代表 1 月到 12 月。

只指定年、月、日而不指定时间时，默认的时间是 0 时 0 分 0 秒。

③　创建指定日期和时间的对象

创建日期对象的同时还可以指定日期和时间。例如：

```
var d_Date = new Date(2001,9,1,12,20,20);
```

这里生成的日期和时间是 2001 年 10 月 1 日 12 点 20 分 20 秒。

(2)　日期对象的方法

获得日期对象的方法有如下几种。

- getYear()：获得日期对象的年份。
- getMonth()：获得日期对象的月份。
- getDay()：获得星期几。其返回值从 0～6，代表星期日到星期六。

设置日期对象的方法有如下两种。

- setYear(年份)：设置日期对象的年份。
- setMonth(月份)：设置日期对象的月份。

(3) 获得与设置时间

JavaScript 可以用 getTime()与 setTime(毫秒数)方法来获取和设置从 1970 年 1 月 1 日 0 时 0 分 0 秒开始的毫秒数。如果 d_Date 日期对象所代表的时间是 1998 年 2 月 1 日 15 时 30 分 27 秒，那么通过 d_Date.getTime()方法获得的值是 886 318 227 000。

4) 数学对象

JavaScript 的数学对象提供了许多强大的数学运算功能，完全能够满足编写脚本中对于数学运算的需要。数学对象与前面几种对象不同，它不需要使用 new 操作符来创建，可以直接使用 Math 来调用数学对象，其属性也就是标准数学常量，其方法构成了数学函数库。所有的属性和方法都是静态的，其使用格式为：Math.属性和 Math.方法。

数学对象中有许多属性，例如 PI，代表圆周率的值，运算时可以直接使用 Math.PI 来获得圆周率。除此之外，Math.SQRT2 代表 2 的平方根，Math.LN2 代表 2 的自然对数。具体可参考表 5.9。

表 5.9 数学对象中的几个常用属性

属　性	说　明
E	欧拉常数，约为 2_718
LN10	10 的自然对数(自然对数以欧拉常数为底)，约为 2_302
LN2	2 的自然对数，约为 0.693
LOG10E、LOG2E	以 10 和 2 为底的欧拉常数 E 的对数，约为 0.434 和 1.442
PI	圆周率常数，约为 3.14159
SQRT1_2	1/2 的平方根，约为 0.707
SQRT2	2 的平方根，约为 1.414

数学对象中有许多方法，现在介绍下面 5 种。

(1) min(值 1,值 2)与 max(值 1,值 2)

min()与 max()都有两个参数，分别代表两个值。min()的功能是返回两个数值中的小者；max()返回两个数值中的大者。例如，min(12,8)返回 8，max(12,8)返回 12。

(2) round(数值)、ceil(数值)和 floor(数值)方法

round()、ceil()和 floor()三个方法都是将参数取整后返回，不同之处是取整的方法不同。round()方法是将参数四舍五入后取整；ceil()方法是返回大于或等于参数的最小整数；floor()方法是返回小于或等于参数的最大整数。例如 round(3.5)返回 4，ceil(3.5)返回 4，floor(3.5)返回 3。

(3) random( )方法

使用 random()方法能够产生 0～1 的一个随机数，当需要用一定范围的数字测试某个项目时，利用它产生数据特别方便。例如想产生 100 个 10～100 的随机数时，可以利用下面的程序：

```
var i;
var ss = new Array(100);
for (i=0;  i<100;  i++) {
ss[i] = Math.round ((100-10) * Math.random()) + 10;
}
```

(4)　sqrt(数值)方法

sqrt()方法的功能是返回数值的平方根。例如，要取 2 的平方根有两种形式，一种是运用 Math.SQRT2 属性，另一种方法就是用 Math.sqrt(2)方法。

(5)　abs(数值)方法

abs()方法的功能是返回数值的绝对值。例如，取−98 的绝对值，可以利用 Math.abs(−98)得到。

除此之外，JavaScript 中可以用 sin(数值)、cos(数值)和 tan(数值)方法获取数值的正弦、余弦和正切函数值。利用 asin(数值)、acos(数值)和 atan(数值)获得反正弦、反余弦和反正切函数值。这里的数值都是代表弧度值。

5.3　多媒体的引用

在网页中可以引用和运行网站内部的多媒体文件，也可以与外部的多媒体文件相连。下面分别讲述这两种方法。

5.3.1　内部多媒体文件的引用

有 4 种标记可以分别用来引用不同类型的多媒体。4 种标记是<img>、<embed>、<object>和<bgsound>。

- <img>标记：利用这个标记不仅可以插入图像，还可以用来播放 Video for Windows 多媒体文件(.avi)。引用的格式是：<img src= "URL">。
- <embed>标记：用来嵌入的多媒体文件包括电影、声音、虚拟现实语言(VRML)等，其格式可以是 MIDI、WAV、AU。使用时需要在浏览器中安装播放相应多媒体文件的插件。引用的格式是：<embed src= "URL">。
- <object>标记：用来插入 ActiveX 控件，还可以插入其他的 OLE 对象，如图像、文档、动画、小程序等，由于它的使用范围广，因此，常被称做“通用标记”。引用的格式是：<object 属性=属性值></object>。
- <bgsound>标记：用来插入背景音乐，WAV、MIDI 等声音文件都可以用它，但一般只适用于 IE，其参数设定不多。格式是：<bgsound src="URL" loop="*">。其中 loop 用来设定循环播放的次数，loop=2 表示重复两次，loop=infinite 代表不断循环直到关闭时为止。

5.3.2　外部多媒体文件的引用

对外部多媒体文件的引用主要通过超链接标记<A>的 HREF 属性进行，其格式为

```
<A HREF="外部多媒体文件名">提示文本</A>
```

5.4 DHTML 的应用示例

下面用 5 个示例由浅入深地介绍 DHTML 中编写脚本的方法。

每个 HTML 元素标记都有很多事件，如鼠标单击(onclick)、鼠标进入(onmouseover)、鼠标离开(onmouseout)等。可以直接在【源】视图窗口中为这些事件编写代码，为此需要先在标记的定义中写明事件，然后在<script>…</script>中编写脚本。也可以利用系统的帮助来简化这个编写的过程。

在【源】视图窗口的上部有两个小的下拉列表框，左边的下拉列表框中将列出已经放进网页的 HTML 元素，右边的下拉列表框列出了该元素的事件。界面如图 5.2 所示。



图 5.2 HTML 元素及其事件窗口

只要在这里选择了 HTML 元素及其事件以后，系统就会给出事件代码的框架，只需在这个框架中编写代码就可以了。

有时，虽然在网页中放进了某个 HTML 元素，但在下拉列表框中找不到它，系统也就不能给出事件代码的框架，这往往是因为还没有给该元素设置 id 属性，只要把这个属性加上去就可以了。

例 5.1 简单的提示。

假定有一个输入文本框(Text1)，必须在其中输入数据，否则当鼠标指针离开文本框时，将提示错误。

先在网页中放入 Input(Text)元素，在【源】视图窗口中选择该元素以及它的 onmouseout 事件，然后编写如下代码。

```
function Text1_onmouseout()
{
  if(Text1.value=="")
    {
     alert("不能为空!");        // 提示
     Text1.focus();            // 将光标的焦点仍然放置到输入文本框中
}
}
```

程序运行中输入为空时的提示如图 5.3 所示。

例 5.2 图片切换。

网页中放入一图片，当鼠标指针移动到图片上面时，自动转换成另一张图片，鼠标指针离开时又还原成原来的图片。



图 5.3　提示输入不能为空

在这个示例中对同一个 HTML 元素要写两个事件，一个是鼠标进入的事件，另一个是鼠标离开的事件，如果系统没有给出事件代码的框架，则需要设置 id，如这里设置的 id=img1。假定原来放入的图片是 picture1.jpg，鼠标进入时转换为 picture2_jpg，鼠标离开时又还原为 picture1.jpg。

完整的代码如下。

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Untitled Page</title>
<script language="javascript" type="text/javascript">
<!--
function img1_onmouseover() {
img1.src=" picture2_jpg ";
}

function img1_onmouseout() {
img1.src=" picture1.jpg ";
}
// -->
</script>
</head>
<body>
    <img id = img1 src="picture1.jpg" language="javascript"
                    onmouseover="return img1_onmouseover()"
                    onmouseout="return img1_onmouseout()" />
  </body>
</html>
```

代码中用<!--…-->注释符是为低版本浏览器(IE 低于 4.0)准备的。因为这些低版本浏览器不支持 DHTML 的脚本。

例 5.3　自动导出数据。

在网页中原始数据当然应该由客户自己输入，而导出的数据最好由系统自行计算。这样可以减少错误。

下面就是一个计算乘法的示例。只需要在前两个文本框中输入值，然后单击第三个文本框，即可显示结果。界面如图 5.4 所示。



图 5.4　自动导出结果

完整的事件代码如下。

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head>
    <title>Untitled Page</title>
<script language="javascript" type="text/javascript">
     function Text3_onclick()
      {
       var s1,s2,s3;
       s1 = eval(Text1.value);
       s2 = eval(Text2_value);
       s3 = s1 * s2;
       Text3.value = s3;
     }
</script>
</head>
<body>
   <input id="Text1" style="width: 96px" type="text" />*
   <input id="Text2" style="width: 96px" type="text" />=
   <input      id="Text3"      type="text"      style="width:      104px"
language="javascript" onclick="return Text3_onclick()" />
       <br />
</body>
</html>
```

例 5.4 图像自动交替地进行淡入、淡出显示。这是一个比较复杂的示例，功能全部由代码来实现。完整的代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/
TR/xhtml11/DTD/xhtml11.dtd">
<html>
<head>
   <title>DHTML 的示例</title>
   <style><!--
     body {font-size:9pt}
     -->
   </style>
<script language="JavaScript">
var flag=0;
//全局变量 flag 用来控制绘出的图像; flag=0 时绘出图像 image2_jpg, flag=1 时则绘出
  image1.jpg
var act;
function Change()
       {
         act=window.setInterval("AutoChange()",9500);
              //每隔 9.5 秒执行函数 AutoChange()
       } //function

function AutoChange()
        {
```

```
            if(flag==0)
             {
               flag=1;
               Img.filters.blendTrans.Apply(); //用 Apply 方法停止原图像的绘出
               Img.src = "image2_jpg";      //将 Img 设置成另一幅图像 image2_jpg
               Img.filters.blendTrans.Play(); //用 Play 方法调用过滤器的淡出效果
              }
            else
               {
                flag=0;
                Img.filters.blendTrans.Apply();
                Img.src = "image1.jpg";
                Img.filters.blendTrans.Play();
               } //if
             }  //function
      </script>
    </head>
    <body onload="Change()">
       <img  id="Img"  src="image1.jpg"  alt=" 我 会 变 换 图 像 ！ "
  style="filter:blendTrans(duration=7)">
    </body>
  </html>
```

代码中 act=window.setInterval("AutoChange()",9500)设置的时间为 9.5 秒，此时间一定要比 style="filter:blendTrans(duration=7)"设置的时间(7 秒)长，否则将产生脚本语言错误。

代码通过过滤器对可见的对象进行过滤以达到动态的效果。CSS 拥有扩充的过滤器，将它与动态 HTML 相结合，可以制造出各种动态的效果。

有两种类型的过渡过滤器：Blend 和 Reveal。

用 Blend 过渡，可以在指定的时间间隔内实现图像的简单的淡出和淡入。其基本句型如下

```
style="filter:blendTrans(Duration=过渡的时间)"  //过渡时间的单位为“秒”
```

用 Reveal 过渡，可以通过使用不同的技术有选择地显示或隐藏图像，它的效果有很多。其基本句型如下：

```
style="filter:revealTrans(Duration=过渡的时间,Transition=过渡的类型)"
                   // 过渡时间的单位为“秒”, Transition 的取值范围是 0～23
```

例 5.5　网页中运行“视频”图像。

如果要在网页中运行视频图像时，先在网站中下载相关文件(例如*.wmv)，然后在选择的位置上编写出如下代码。

```
<embed  src="image\ 文 件 名 .wmv"  autostart="true"  loop="*"  width="300px"
height="300px">
</embed>
```

5.5 小　结

DHTML 是浏览器端的动态技术，它能使下载后的网页中的元素动起来，从而大大增强网页的生动性和吸引力。而这一切都是利用浏览器本身的资源获得的，没有增加服务器的负担和网络上的流量，这是 DHTML 的最大特点，也是它的最大优点。

DHTML 不是一种单一的技术，而是多种技术的综合，可以用一个简单的公式来表达：

```
DHTML = DOM + HTML 4.0 以上 + CSS + 事件 + 脚本 +多媒体
```

DHTML 技术具有如下局限性。

- 它只能使元素在形式(位置、形状等)上产生变化，当需要获得变化的数据，例如从数据库中读出更新后的数据时，不得不依靠服务器端的技术来完成。
- DHTML 技术受限于浏览器本身，而随着 Internet 的客户日益多样化、小型化，浏览器端的功能可能会变得越来越弱。

这一切说明网站开发的重点是服务器端技术，ASP.NET 是基于服务器的技术，但是如有可能(如 IE 的版本在 4.0 或以上时)，仍应该注意发挥 DHTML 的辅助功能。这方面的相关内容将在后面章节中讲述。

5.6 习　题

1. 填空题

(1) DHTML 的设计思想是：浏览器从服务器端下载文档后，利用_________的资源，在不增加________端负担和网上传输流量的前提下，使网页的某些元素“动”起来。

(2) DHTML 不是一种单一的技术，而是多项技术的综合。在文档对象模型(DOM)的基础上，包括__________、__________、___________、__________等技术。

(3) 文档对象模型(DOM)是英文__________________的缩写，它是________的基础。

(4) JavaScript 是由 Netscape 公司开发的一种解释型语言。JavaScript 既可在_______又可在_________端解释执行，JavaScript 是一种_______面向对象和事件驱动的跨平台的语言。

2. 简答题

(1) 你对 DHTML 是如何理解的？它与 DOM 有什么关系？

(2) HTML 常应用于多媒体的标记有哪些？

3. 操作题

自行查找有关 DHTML 方面的资料，完成以下内容。

(1) 做一个漂亮的数字或模拟时钟。

(2) 做一个能在页内移动且和四周碰撞反弹的广告图片。

第 6 章　利用 jQuery 设计动画

在上一章讲述 DHTML 技术的时候，只列举了几个简单的动画示例。实际上很多网页中的动画常常要比这些示例丰富得多。一张好的网页实际上就是一幅优美的艺术品。

面对这一张张优美的“艺术品”，作为网页的设计者常常会问自己，如何才能设计出如此高水平的动画来。

应该说，这并不是一件容易的事。因为动画的设计离不开 JavaScript 语言，而掌握 JavaScript 语言是大多数网页设计者遇到的难题。为了突破这个难题，近几年来，软件界出现了几种不同的 JavaScript 语言库，用来简化对 JavaScript 语言的应用，其中比较突出的有以下几种版本：Prototype、Dojo、YUI、Ext JS、MooTools、jQuery 等。这几种版本从不同的角度出发，提出了不同的解决方案。

jQuery 是以 John Resig 为首，一批具有奉献精神的顶尖的 JavaScript 专家组成的开发团队，于 2006 年开发出来的系统，它大大简化了 JavaScript 语言的应用，利用这个库，能用简短的代码实现较复杂的功能(他们的口号是：Write Less, Do More)。该系统采取开源方式，允许设计者免费下载使用。目前已经广泛应用于各类主流网页的设计之中。如 PHP、ASP.NET、JSP、Ruby、Dreamweaver 等网页中都采用了 jQuery 技术。

微软已于 2008 年公开宣布全面支持 jQuery 技术，并将该技术无缝地嵌入到 ASP.NET 3.5 和 ASP.NET 4.0 的系统中。

jQuery 是一个开放源代码的工具包，可以用来处理一些常见的客户界面任务。主要包括以下三个方面。

- 网页动画设计。
- 对网页表单以及校验的支持。
- 对 Ajax 技术的支持。

由于篇幅限制，本书只集中讲述 jQuery 用于动画设计方面的内容。下面将先讲解与 jQuery 相关的基础知识，再讲解用 jQuery 设计动画的方法。通过学习不仅能够提高网页动画的设计能力，还能加深对 DOM、HTML、CSS、DHTML 等概念的理解。

6.1　jQuery 基础

6.1.1　什么是 jQuery

jQuery 是一个优秀的 JavaScript 语言库，目的在于简化对 JavaScript 语言的应用。简化的方法是将多条 JavaScript 语句封装为函数，再将函数分类抽象成库。使用时，直接调用库中的函数即可实现比较复杂的功能。

以设计一幅“淡入淡出的图像”为例，如果直接用 JavaScript 语言来设计，需要缩写近 20 条代码(见例 5.4)。现在用 jQuery 来设计，只需使用 1～2 句函数调用即可。函数调用的语句是：

```
$("img").hide(3000);   // 用 3 秒时间逐步隐藏动画(淡出)
```

或者

```
$("img").show(3000);   // 用 3 秒时间逐步显示动画(淡入)
```

通过这个简单的示例可以看出，jQuery 语言的主要特点是语句简短且易读性强，更为重要的是，它是从多条 JavaScript 语句中抽象出来的，经过了反复验证，与使用多条原始 JavaScript 语句进行设计相比，出错的概率大大降低，语句变得更加“健壮”。

6.1.2　jQuery 能做什么

jQuery 的主要任务就是对浏览器中 DOM 元素进行操作，以实现客户与浏览器之间的交互。其具体任务包括：

- 获取网页元素。
- 修改网页外观。
- 修改网页内容。
- 给网页设计动画。
- 客户与浏览器进行交互。
- 以局部刷新方式与服务器交换信息。
- 简化其他 JavaScript 的任务。

6.1.3　JQuery 的特点

概括地说，JQuery 具有以下几个特点。

- 使用 html、CSS 等通用标准。
- 为各类浏览器通用。
- 总是以集合方式工作。
- jQuery 库的容量非常小，代码紧凑。
- 大量成熟的插件供设计者调用。
- 与 JavaScript 语言以及其他 JavaScript 库相互兼容。

6.1.4　配置 jQuery 的使用环境

使用 jQuery 前需要配置使用环境，为此需要首先从因特网中下载 jQuery 库，为了下载当时的最新版本，打开 jQuery 的官方网站，其网址是 http://jQuery.com/。该网站的主页面如图 6.1 所示。

这是 2010 年 12 月 jQuery 的版本情况。从网页右下方可以看出，jQuery 当时的最新版本(Current Release)是 v1.4.3。客户可以根据需要选用两种下载版本之一。

- PRODUCTION(jQuery-1.4.3_min.js)：是压缩版本，其容量仅为 26KB，主要用于产品开发。
- DEVELOPMENT(jQuery-1.4.3_js)：是无压缩版本，容量为 179KB，包括更多的源代码和相关文件，主要用于系统测试和开发。



图 6.1 jQuery 的官方网站

现在准备下载前一种版本。先选择版本前面的■，再单击 Download 按钮，将该库下载到指定的目录下来。解压后再放置到网站的目录下来(例如放在网站目录 scripts 下面)，并在<head>…</head>字段中引用 jQuery 库。引用的代码如下。

```
<script type="text/javascript" src="scripts/jQuery.min.js"></script>
```

6.1.5 jQuery 的起始语句

jQuery 的主要任务就是读取和处理浏览器中的 DOM 元素，为此 jQuery 的每个脚本前都要写一段“起始语句”，以保证只有在 DOM 文档载入浏览器以后才执行其他代码。起始语句的格式如下。

```
$(document).ready(function() {
      // 后续执行的代码(语句体)
});
```

语句中的$在 jQuery 中用得最多，它是 jQuery 的缩写。由于代码中经常要用到 jQuery，用$代替能够缩短代码的长度。如：$("#foo")与 jQuery("#foo")是等价的。

在起始语句中，系统用一个 ready()事件作为处理 DOM 文档的开始，它表示只有将 DOM 文档调入浏览器以后才开始执行后续语句(这些后续语句又称为“语句体”)。

对起始语句还可以作进一步的简化。简化后的格式如下。

```
$(function(){
  // 后续执行的代码(语句体)
  });
```

在 JavaScript 语言中，其实也有一个类似的方法，名为 window.onload。但该方法的执行过程与这里的起始语句稍有区别。window.onload 方法是在网页中所有元素(包括各种关联文件)都完全加载到浏览器以后，才开始执行后续语句。如果有一个大型图库网站，必须等到所有文件(包括全部图片)都下载完后，才能执行后续语句。而执行$(document).ready()方法时，只需等到 DOM 下载完毕即可执行后续语句。从时间上看后者的启动时间会短一些。

6.1.6 示例

下面用一个简单的示例来概括一个 jQuery 程序所包括的内容。程序代码如下。

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
        <title></title>
        <!— 引用 jQuery 库 →
        <script type="text/javascript" src="scripts/jQuery-
        1.4.3.min.js"></script>
         <!—jQuery 语句→
        <script type="text/javascript">
         <!—起始语句 →
           $(document).ready(function(){
             <!— 后续语句(语句体) →
              $("a").click(function(){
                 alert("您好！");
             });
           });
        </script>
  </head>
  <body>
        <form id="form1" runat="server">
          <!—HTML 标记→
         <div>
           <a>一个最简单的程序</a>
         </div>
        </form>
    </body>
</html>
```

运行本示例，并且单击链接标记(<a>…)中的提示时，将弹出“您好！”的问候窗口。在这个程序中语句体实际上只做了两件事。

- 给 HTML 元素定位。如程序中的$("a")语句。
- 处理事件。如处理 click()事件。

这是一个最简单的 jQuery 程序，整个程序涉及 HTML 标记、CSS 定义(本示例没有用到)、引用 jQuery 库、起始语句以及后续执行语句(语句体)5 个部分。其中引用 jQuery 库以及起始语句虽然不可缺少，但都比较固定。因此设计的重点是 HTML 标记、CSS 定义与语句体三者之间的结合。

6.2　jQuery 对元素定位

6.2.1　定位集合对象

jQuery 的定位方式与 CSS 的定位方式相同，即通过“选择器”来定位元素。定位的结果是选择了一组 jQuery 对象集。jQuery 的定位语句格式如下。

- $("标记名")：根据 HTML 标记定位。例如，$("a")，定位所有的标记为 a 的元素。
- $("#元素 id")：根据元素 id 定位。例如，$("#content")，定位 id=content 的元素。
- $(".class 名")：根据类名定位。例如，$(".lend")，定位所有的类名为 lend 的元素。
- $("标记 1,标记 2, …")：定位多标记元素。例如，$("td,p,div")，定位所有标记为 td、p、div 的元素。

- $("父元素 后代元素")：定位后代元素。例如，$(".link a")，定位类名为 link 后代中所有的 a 元素。
- $("*")：定位所有元素。

jQuery 对象集是 jQuery 特有的一种类型，它代表一组 jQuery 对象的集合。而原始 JavaScript 的定位语句定位的是单个 DOM 对象。例如：

```
var  node=getElementById("元素 ID");
```

或者

```
var  node=getElementBydName("元素名");
```

上述语句中的 node 都是单个 DOM 对象。jQuery 对象集与 DOM 对象不同，各自只能使用属于自己的方法(第 5 章中 JavaScript 的方法只适用于 DOM 对象)。当然集合对象与单个 DOM 对象之间也很容易相互转换。转换方法如下。

(1) 从 jQuery 对象转换成 DOM 对象时，有如下两种情况。

① jQuery 是一个数组对象，可以通过[index]方法得到 DOM 对象。例如：

```
var $content = $("#content");
```

其中$content 是 jQuery 对象集。变量名中的$并不是必要的，放在这里只是为了与一般变量区别。

```
var content = $content[0];
```

其中 content 是从 jQuery 对象集转换成的单个 DOM 对象。

② 通过 get(index)方法获得 DOM 对象。例如：

```
var $content = $("#content");          // $content 是 jQuery 对象集
      var content = $content.get(0);   // content 是转换后的第一个 DOM 对象
```

(2) 从 DOM 对象转换成 jQuery 对象时，只需用$()将 DOM 对象包装起来即可。例如：

```
var content = document.getElementById("content")   //得到 DOM 对象
var $content = $(content);                         //转换为 jQuery 对象
```

下面用一个动画示例，来演示 jQuery 的定位方法。这是一个同时操作 4 张图片的程序，其显示的界面如图 6.2 所示。当单击右下方的 button 按钮时，4 张图片同时在 3 秒内隐藏；若再单击按钮时，在 3 秒钟内又逐步恢复原状。









button

图 6.2 图片隐藏/显示示例

下面是 jQuery 的程序代码，代码的后面均用注释语句加以说明。

```
    <script type="text/javascript">
        $(document).ready(function() {        //起始语句
            $("#Button1").click(function() { //触发事件
            if ($("img").is(":visible")) {    //判断语句
                $("img").hide(3000);          //语句体，3 秒内逐步隐含
                } else {
                $("img").show(3000);          //语句体，3 秒内逐步恢复显示
                }
            });
        });
    </script>
</head>
<body>
    <form id="form1" runat="server">
    <div>
      <img alt="0 号图片" src="images/图片小 0.jpg" />
      <img alt="1 号图片" src="images/图片小 1.jpg" />
      <img alt="2 号图片" src="images/图片小 2.jpg" />
      <img alt="3 号图片" src="images/图片小 3.jpg" />
      <input id="Button1" type="button" value="button" />
    </div>
    </form>
</body>
</html>
```

语句体中使用了两种定位语句。

- $("#Button1")：定位所有 id=Button1 的元素，用来与后面的事件绑定。在本例中只有一个 Button1。
- $("img")：定位所有 HTML 标记为 img 的元素，它是一组元素的集合。在本例中有 4 个 img 标记元素(即 4 张图片)，它们是事件的执行者。在这里是 4 张图片一同隐藏或者一同显示。

6.2.2　定位单个(或部分)对象

有时只希望访问某个特定元素(或部分元素)，jQuery 也为此提供了很多简便的方法。

下面使用的是一种自定义选择器的方法，其格式是在选择器后面增加一个冒号(：)再加上自定义名。

- :even：序号为偶数的元素。如$("img:even")。
- :odd：序号为奇数的元素。如$("img:odd")。
- :first：首位元素。如$("img:first")。
- :last：最后的元素。如$("img:last")。
- :eq(index)：序号为 index 的元素。如$("img:eq(2)")，代表第三个元素(序号从 0 开始)。

现在对前面的 jQuery 程序进行改写，只将奇数的图片进行“隐藏/显示”。代码修改如下。

```
<script type="text/javascript">
        $(document).ready(function() {
            $("#Button1").click(function() {
            if ($("img:odd").is(":visible")) {
                $("img:odd").hide(3000);
                } else {
                $("img:odd").show(3000);
                }
            });
        });
</script>
```

运行程序并查看运行的结果。

6.2.3 根据层次关系对元素定位

根据层次关系对元素定位是 jQuery 常用的方法。

- $("ancestor descendant")：选取 ancestor 所有的后代元素(descendant)。例如，$(".link a")代表选取类名为 link 后代中所有的 a 元素(注意中间用空格分开)。
- $("parent>child")：选取 parent 父元素的所有子元素(child)。例如，$(".link>a")代表选取类名为 link 的儿子中所有 a 元素。

6.2.4 在遍历 DOM 中进行定位

在 DOM 的遍历中给元素定位，是 jQuery 常用的方法。DOM 元素之间存在着一种层次关系。其部分结构如图 6.3 所示。

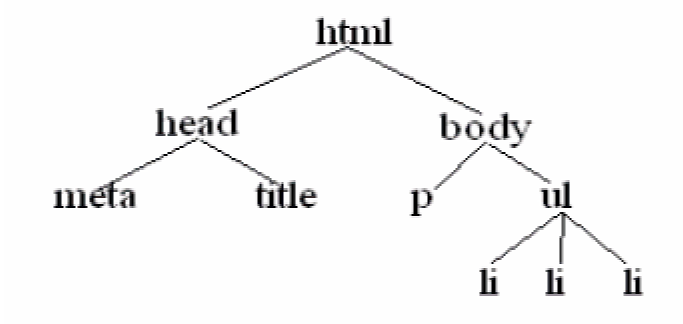


图 6.3 DOM 结构示例

DOM 元素的这种层次关系与人们的家族关系类似，因此可以借助家族之间的称呼来进行类比。以图 6.3 为例，<html> 是所有元素的祖先。换句话说，所有其他元素都是<html>的后代子孙。<head> 和 <body>是 <html> 的两个孩子，<head> 与 <body>之间是兄弟。<p>元素是<body>的孩子，也是<html>的后代，<p>与<ul>元素之间是兄弟等。

- children()方法：利用 children()方法可以获得本元素孩子元素的个数。
- next()方法：用于匹配本元素的下一个兄弟元素。
- prev()方法：用于匹配本元素的上一个兄弟元素。
- siblings()方法：用于匹配本元素前后所有的同辈(兄弟)元素。

6.3　事件与方法

6.3.1　事件

1. 概述

1)　事件的类型

jQuery 的事件包括有多种类型。

- 与鼠标相关的事件有 mouseover、mouseout、focus、blur、mousedown、mouseup、mousemove 等。
- 与键盘相关的事件有 keydown、keypress、keyup 等。
- 其他相关的事件有 load、unload、error、change、select、submit 等。

还可以有一些自定义的事件。

2) 与事件绑定的方法

例如：

```
$("定位元素").click(…);        //将鼠标单击事件绑定到定位元素上
```

再如：

```
$("定位元素").mouseover(…);  //将鼠标进入(覆盖)事件绑定到定位元素上
```

2. 示例

下面是一个对树形控件设计的示例。控件中包括几个列表项，当单击其中某项的名称时，该项的子项显示或者隐藏起来。

1)　HTML 的相关代码

HTML 的相关代码如图 6.4 所示。

2)　对应的 DOM 结构图

上述代码对应的 DOM 结构如图 6.5 所示。

3)　jQuery 的代码

jQuery 的代码如下。

```
<script type="text/javascript">
       $(function() {
           $(".m-treeview li span").click(function() {
               var $ui = $(this).siblings("ul");
               if ($ui.is(":visible")) {
                   $ui.hide();
               } else {
                   $ui.show();
               }
               return false;
           });
       });
       </script>
```

```
<form id="form1" runat="server">
  <ul class="m-treeview">
    <li class="m-expanded">
      <span><a href="#">衬衫</a></span>
      <ul>
        <li><a href="#">无袖衬衫</a></li>
        <li><a href="#">短袖衬衫</a></li>
        <li><a href="#">长袖衬衫</a></li>
      </ul>
    </li>
    <li class="m-expanded">
      <span><a href="#">裙装</a></span>
      <ul>
        <li><a href="#">长裙</a></li>
        <li><a href="#">短裙</a></li>
        <li><a href="#">超短裙</a></li>
      </ul>
    </li>
    <li class="m-expanded">
      <span><a href="#">外衣</a></span>
      <ul>
        <li>...</li>
      </ul> ...
    </li>
  </ul>
</form>
```

图 6.4 树形控件代码的层次关系

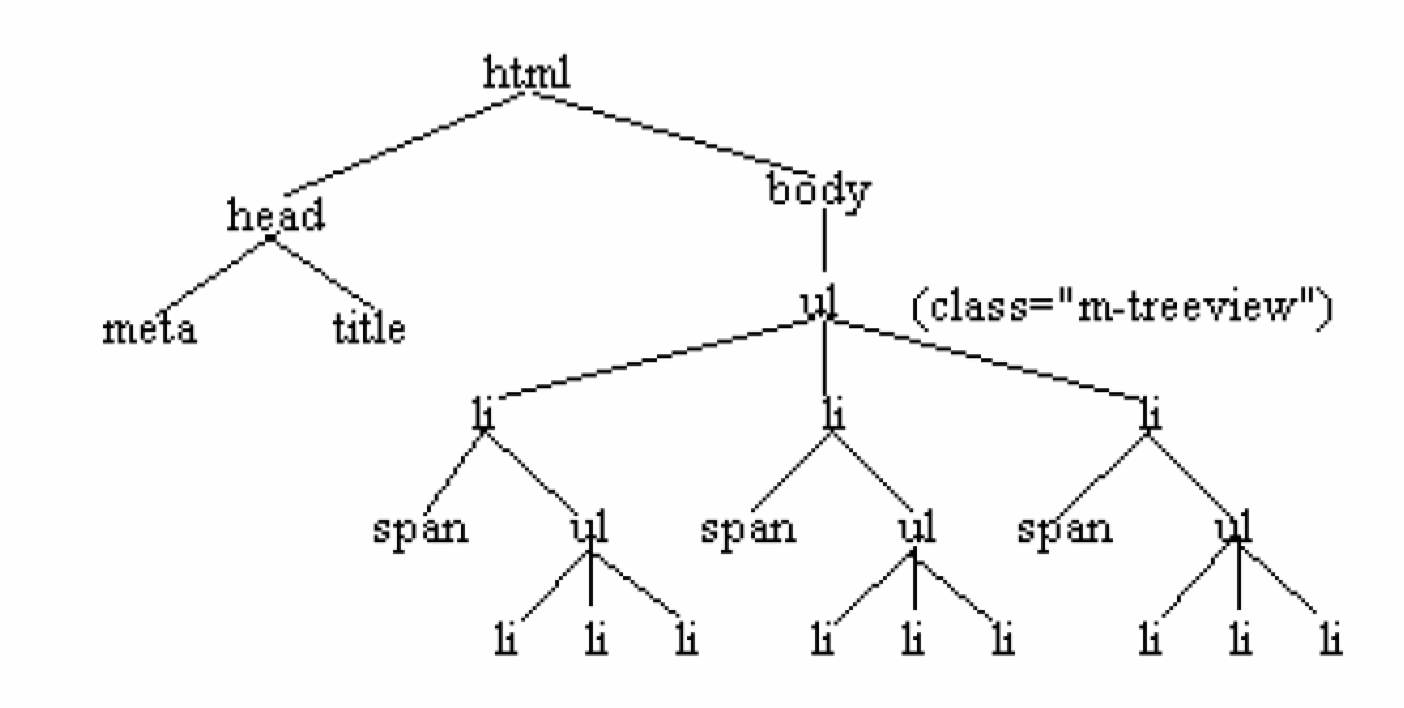


图 6.5 树形控件的 DOM 结构简图

4) 运行结果

运行结果如图 6.6 所示。

- 衬衫
- 裙装
  - 长裙
  - 短裙
  - 超短裙
- 外衣

图 6.6　树形控件运行结果

5)　代码分析

对于 jQuery 语句体的代码分析如下。

- $(".m-treeview li span")是根据层次定位的语句，目的是找到类名为 m-treeview 后代标签 li，在后代标签 span 中的元素集合，用来绑定 click 事件。
- var $ui = $(this).siblings("ul");等号右边的$(this)是 jQuery 中一个重要的关键字，它是单个 DOM 对象(不是对象集)，通常代表被单击的对象。在本例中代表被单击的 span 元素。整个语句连起来的意思就是，将被单击的 span 元素的所有兄弟元素取出来，赋给$ui 变量。

  这里的赋值实际上是一种数据缓存的办法，如果程序中某些元素要多次使用时，可以先将其缓存起来，以后直接调用变量名即可，用不着每次重新定位，也就提高了运行的效率。
- if ($ui.is(":visible"))语句用来判断被缓存的变量($ui)是否显示，以便决定是执行隐藏(hide())还是显示(show())操作。

6.3.2　成对事件代码的简化

有些事件往往成对地出现，如 hover() 与 toggle()，系统对这些事件提供了简化代码的方法。

1. hover()方法

hover()方法的语法结构如下。

```
hover(enter, leave);
```

该方法包括两个函数，中间用逗号(,)隔离。鼠标进入时执行第一个函数；鼠标退出时执行第二个函数。格式如下。

```
<script type="text/javascript">
       $(function() {
           $("元素定位").hover(function() {
              // 鼠标进入时的函数体
           }, function() {
              // 鼠标退出时的函数体
           });
       });
</script>
```

2. toggle()方法

toggle()方法的语法结构如下。

```
toggle(fn1,fn2,…fnN);
```

当第一次单击时触发第一个函数(fn1)，再次单击同一元素时，触发第二个函数(fn2)，其他依次触发。格式如下。

```
<script type="text/javascript">
       $(function() {
           $("元素定位").toggle(function() {
               // 第一个函数
               },function() {
               // 第二个函数
             });
         });
    </script>
```

6.3.3 事件冒泡

1. 什么是冒泡

由于在 HTML 内有些元素呈嵌套状态，这种情况下，内层元素被某个事件触发时，其外层元素也会响应同一事件，其响应过程从内向外逐步扩散，如同水中的气泡从下而上移动一样。

2. 停止冒泡

事件冒泡可能带来某种意想不到的结果，因此有必要对事件的作用范围进行限制。系统提供的 stopPropagation()方法可以用来停止冒泡，也可以用 return false 语句来停止冒泡。

3. 阻止默认行为

网页元素通常都有自己的默认行为。例如单击超链接标记时将跳转到新网页；单击【提交】按钮时提交表单内容等。可以用 preventDefault()方法来阻止元素的默认行为，同样也可以用 return false 来阻止默认行为。

6.4 对节点的操作

6.4.1 创建新节点

通过 jQuery 可以给 DOM 增加新的临时节点。方法是首先定义新节点的内容，然后再用 append()方法将新节点插入到 HTML 中。

本示例是 6.3.1 节示例的继续，目的是给树形节点的列表增添新节点。jQuery 的代码如下。

```
$(function() {
```

```
$(".m-treeview li span").click(function() {
  var $ui = $(this).siblings("ul");
  var $newnod1 = $("<li title='连衣裙'> 连衣裙 </li>"); //创建第一个新节点
  var $newnod2 = $("<li title='吊带裙'> 吊带裙 </li>"); //创建第二个新节点
      $ui.append($newnod1);              // 将新节点插入 HTML 中
      $ui.append($newnod2);
    });
  });
```

增加新节点后的界面如图 6.7 所示。

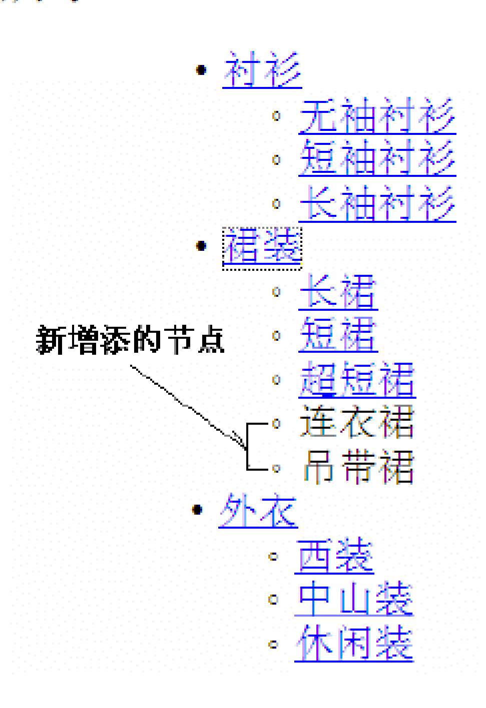


图 6.7　增加节点示例

6.4.2　删除节点

删除节点时使用 remove()方法。

例如：

```
$("ul li:eq(1)").remove();    //代表将 ul 后代第二个 li 元素清除
```

6.4.3　复制节点

复制节点也是 jQuery 对 DOM 的常用操作之一。复制时使用 clone()方法。例如：

```
$(function() {
   $("ul li").click(function() {        //单击 ul 下面的某项 li 时
      $(this).clone().appendTo("ul");  //将该项复制并增添到 ul 下面
      });
   });
```

按照上述方法复制后的新元素并不具备任何行为。如果需要新元素也具有复制功能(本例中是单击事件)应将代码

```
$(this).clone().appendTo("ul");
```

改写为

```
$(this).clone(true).appendTo("ul");
```

即在 clone()方法的括弧中增加了 true。这样，被复制的新元素也同样具有被复制的功能了。

6.5 样 式 操 作

jQuery 可以通过样式操作来临时增添或移除元素的 CSS 定义，以改变元素的外观。

6.5.1 获取样式

获取元素样式属性的语句如下。

```
var p_class=$("p").attr("class");  //获取<p>元素中的 class 选择器
```

6.5.2 增加样式

为匹配的元素增加 class 选择器。其语句如下。

```
$("元素名").addClass("类选择器名");
```

例如：

```
$("p").addClass("myclass");
```

结果是

```
<p class="myclass">…</p>
```

允许给匹配的元素增添多个类，类名之间用空格分开。例如：

```
$("p").addClass("myclass1 myclass2");
```

结果是

```
<p class="myclass1 myclass2">…</p>
```

6.5.3 删除样式

为匹配元素删除 class 选择器的语句如下。

```
$("p").removeClass("anotherclass");
```

给匹配的元素删除指定的类(这里是指名删除类 anotherclass)。
若不指定类名时，代表删除匹配元素中所有的类选择器。

6.5.4 判断元素是否含有某样式

判断元素是否含有某样式的语句如下。

```
$("p").hasClass("anotherclass"); // 判断<p>元素中是否含有 anotherclass 类选择器，
                                    如果含有该样式时将返回 true
```

6.6　利用 jQuery 设计动画

6.6.1　show()与 hide()方法

show()与 hide()是动画中两个最基本的方法，在 HTML 文档中，为一个元素调用 hide()方法，会将该元素的 display 样式改为 none，此时，系统会记住元素原来的 display 属性。当执行 show()方法时，就会恢复原来的状态。

当 show()与 hide()方法中不带参数时，显示或隐含都会立即执行。如果希望按照一定的速度逐步执行时，可以在参数中增添数字。例如：

```
$("element").show(1000);//将使元素("element")在 1 秒钟(1000 毫秒)内慢慢显示出来
```

同样：

```
$("element").hide(2000);//将使元素("element")在 2 秒钟(2000 毫秒)内慢慢消失
```

也可以用 slow、normal、fast 字符串来指定速度，三者分别代表的时间为 600、400 和 200 毫秒。例如：

```
$("element").show("slow");
```

后面几种方法指定速度的方法与此相同。

6.6.2　fadeIn()与 fadeOut()方法

fadeIn()与 fadeOut()方法只改变元素的不透明度。fadeOut()会在指定时间内逐步降低元素的不透明度，直到元素完全消失。fadeIn()与上述情况相反。

6.6.3　slideUp()与 slideDown()方法

slideUp()与 slideDown()方法用来改变元素的高度。如果一个元素 display 的属性为 none，调用 slideDown()方法时，该元素会由上至下逐步延伸显示。slideUp()的情况与此相反。

6.6.4　animate()方法

animate() 函数功能很强，是动画代码的核心，它可以用于更改随时间而逐步改变的 CSS 样式值。比如可以逐步改变高度、宽度、不透明度和位置等。还可以直接用毫秒或者用慢速(slow)、中速(normal)或快速(fast)等来指定改变的速度。animate()的语法结构如下。

```
animate(params, [speed], [callback]);
```

其中的参数介绍如下。

- params：一个包括样式属性及值的映射，其值只代表执行完成后的结果，起始值就是当前值，不需设置。
- speed：速度参数，可选。例如：

```
animate({opacity: "0.1", left: "+=400"},2000)
```

代表某元素在 2 秒钟时间内，不透明度从当前状态变成 0.1，左边距增加 400px。

- callback：回调函数，即动画执行完成时调用的函数。可选。

6.6.5 示例

下面是一个“飘动的方块”的示例，用来演示利用 animate()方法产生的动态效果。该方块的初始状态如图 6.8 所示。

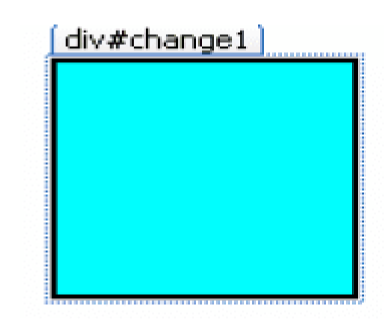


图 6.8 飘动方块的初始状态

1. HTML 代码

对应的 HTML 代码如下。

```
<form id="form1" runat="server">
    <div>
    <div id="square1"></div>
     </div>
    </form>
</body>
</html>
```

对应的 CSS 定义如下。

```
  <style>
    #square1
    {
        width:120px;
        height:110px;
        background-color:Blue;
        position:relative;
    }
   </style>
```

注意：这里的布局采用了“相对定位”方式(即 position:relative)，只有这样才能使方块飘动起来。

2. JQuery 代码

JQuery 代码如下。

```
$(document).ready(function() {
  $("#square1").click(function() {
  $(this).animate({opacity: "0.1", left: "+=400"},2000).animate({opacity:
"0.4", top: "+=160", height: "20", width: "20"},
"slow").animate({opacity: "1", left: "0", height: "100", width: "100"},
```

```
"slow").animate({top: "0"}, "fast").slideUp().slideDown("slow");
  // 在 animate()中用 slow、fast 等来表示速度时要加引号
// 如果用数字表示速度不需加引号
 });
});
```

从本示例可以看出，如果对同一元素使用多种方法时，可以用连写的方式调用方法。此时前一个方法结束时的状态，就是后一个方法的起点。其参数和坐标都不需要重新定义，这将大大简化设计，减少代码的长度。

6.6.6　停止元素动画

很多时候需要停止匹配元素正在进行的动画，此时可以使用 stop()方法。stop()方法的结构是

```
stop([clearQueue][,gotoEnd]);
```

参数 clearQueue 和 gotoEnd 都是可选的，其值为 boolean 型(true 或 false)。clearQueue 代表是否要清空未执行完的动画队列，gotoEnd 代表是否直接将正在执行的动画跳到末尾。

如果直接用 stop()方法(不带参数)，则立即停止当前的动画，并立即执行队列中的下一个动画。

6.7　动画设计示例

下面用两个示例来综合讲解动画的设计方法。

例 6.1　移动的图片。

显示的界面如图 6.9 所示。

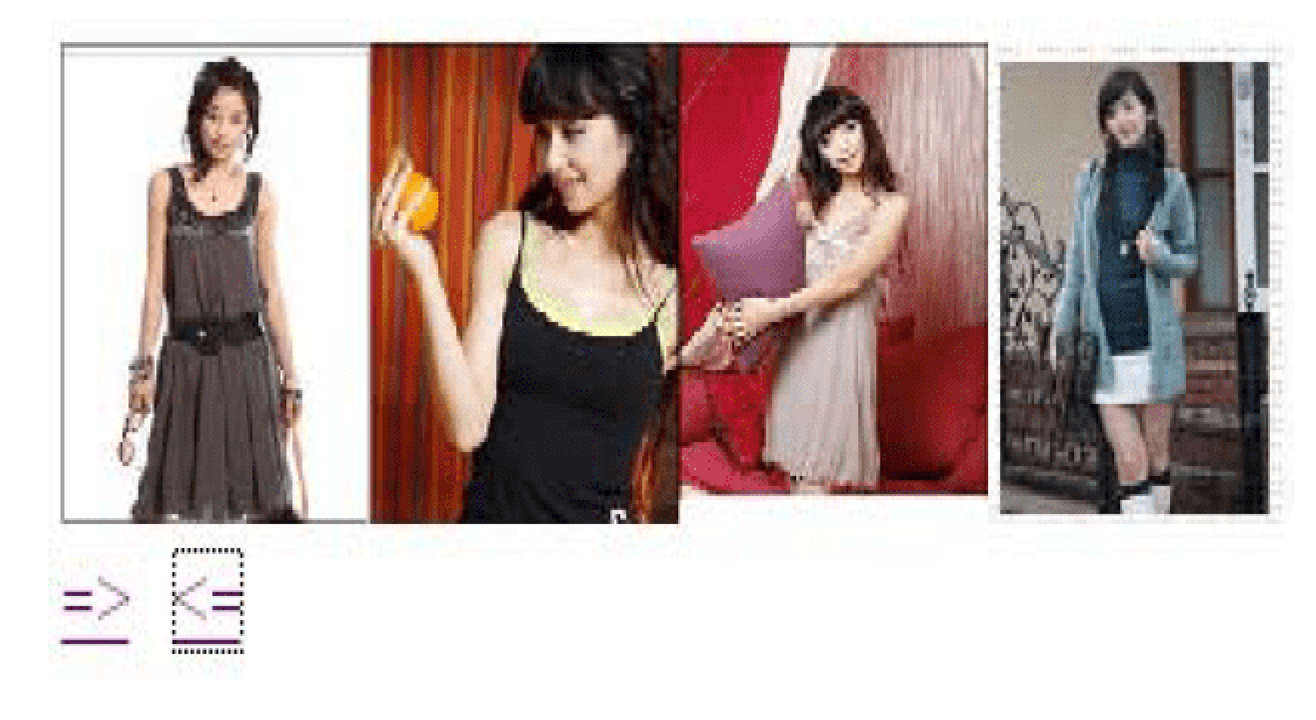


图 6.9　移动图片示例

当单击左下方的按钮时，图片将向左或向右移动，每次只显示其中一(或两)幅图片。

设计过程如下。

(1)　下载并引用 jQuery 库。

(2)　下载图片并进行布局。这里用
< li>列表进行布局。代码如下。

```
<div class="alllist">
    <ul>
     <li><a href="#"> <img alt="image1" src="image/图片小1.jpg" /></a></li>
     <li><a href="#" > <img alt="image2" src="image/图片小2_jpg" /></a></li>
     <li><a href="#"> <img alt="image3" src="image/图片小3.jpg" /></a></li>
     <li><a href="#"> <img alt="image4" src="image/图片小4.jpg" /></a></li>
     <li><a href="#"> <img alt="image5" src="image/图片小5.jpg" /></a></li>
     <li><a href="#" > <img alt="image6" src="image/图片小6.jpg" /></a></li>
   </ul>
  </div>   </div>
 </form>
```

(3) 编写 CSS 定义。

默认情况下网页中每一行将显示一张图片。为了将多张图片排列成一行，应采用“float:left;”方式布局，即从左到右按序排列。CSS 的定义如下。

```
ul li {
     float:left;    // 将图片从左到右排列
}
```

为了允许整个图片移动，将 class=“alllist”定义成“position: relative”布局。

```
.alllist {
      position:relative;  //允许移动
      width:600px;
      height:100px;
 }
```

为了各图片显示的大小一致，CSS 的定义语句如下。

```
   img {
     border:0;
     width:100px;  //定义图片的大小
     height:100px;
}
```

(4) 设置移动按钮。

设置移动按钮的代码如下。

```
<h5 class="moveto"><a href="#"
class="move_to_right">=></a>   
<a href="#" class="move_to_left"><=</a></h5>
```

将其设置为 relative 布局方式，然后将其拖动到合适的位置。

(5) 编写 jQuery 代码。

jQuery 的代码如下。

```
$(function() {
    $(".move_to_right").click(function() {
       $(".alllist").animate({ left: '+=' + 100 }, "slow");
        });
      $(".move_to_left").click(function() {
        $(".alllist").animate({ left: '-=' + 100 }, "normal");
     });
});
```

(6)　限制图片的显示。

如果每次只显示 1～2 张图片时，可以在<div class="alllist">…</div>的外面再加上一个“外框”，并将外框放置在某个固定位置上，对超出外框的图片实现隐藏。其设计思路如图 6.10 所示。

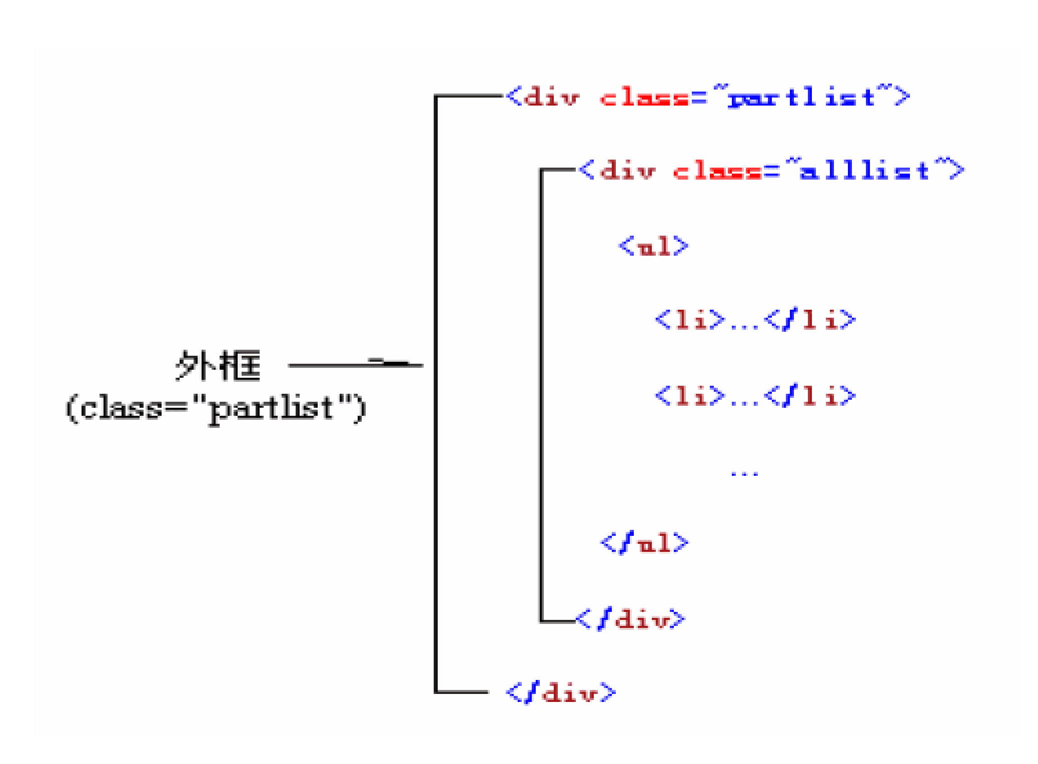


图 6.10　增添外框限制显示范围

CSS 定义的代码如下。

```
.partlist{
          position:absolute;            //用绝对定位的方式进行布局
          top:0;
          left:300px;
          width: 100px;                 //显示一张图片
          height: 100px;
          border: solide 1px blue;      //增加边框
          overflow:hidden;              //隐含超出部分
       }
```

其中 overflow:hidden;属性的作用是将超出外框的图片隐藏起来。

例 6.2　多张图片自动转换。

本示例仍然是逐步显示多张图片，但采用另一种方法设计，设计思路更巧妙，效果更好。设计的要点如下。

- 多张图片像“扑克牌”一样重叠摆放。
- 用 DOM 遍历方式选择图片，用改变纵深序号(z-index)的方法改变显示的图片。
- 当显示完最后一张图片后，自动转向第一张图片继续循环。
- 定时触发上述行为。

显示的界面如图 6.11 所示。



图 6.11　重叠摆放后的显示

相关代码如下。

(1) HTML 布局。

```
<body>
  <form id="form 1 runal="server">
  <div id="slideshow">
    <a href="current">
      <a href="#" ><img alt="人物 1" src="image/r1.jpg"/></a>
    </div>
    </div>
      <a href="#" ><img alt="人物 2" src="image/r2.jpg"/></a>
    </div>
    </div>
      <a href="#" ><img alt="人物 3" src="image/r3.jpg"/></a>
    </div>
  </div>
  </form>
</boby>
```

它表示在类(.slideshow)的下面有三个由 div 组成的层(也可以更多)，每个层上面放有一张图片。上述代码对应的 DOM 结构如图 6.12 所示(这里只画了相关部分)，图中还显示出 jQuery 的遍历过程。

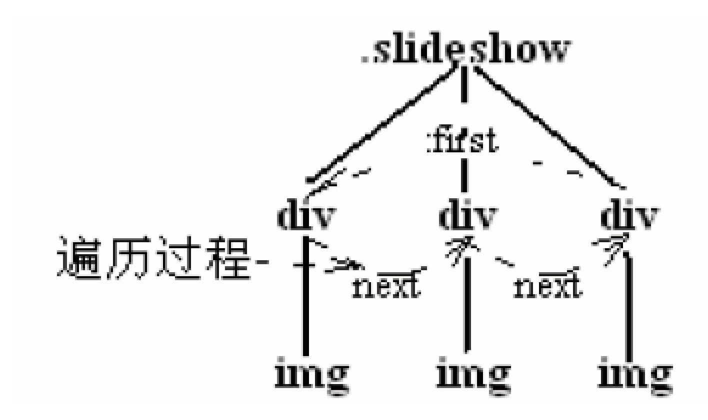


图 6.12 在 DOM 中遍历的过程

(2) CSS 定义。

```
<style type="text/css">
    #slideshow{position:relative;height:50px;width:70px;margin:0 auto 5px ;}
    #slideshow div{
      position :absolute;top:0;left:0;z-index:8;height:50px;width: 70px;
      border:2px solid #0000ff; overflow :hidden;background-color:#FFF;
      }
    #slideshow div img{
      position:absolute;top:0px;left:0px;
      display:block;border:0;margin-bottom:0px;
      }

     #slideshow div.current{z-index:10;}
  </style>
```

定义中将最外部元素采用相对布局(position:relative)方式，以便设计时调整其位置。

内层(即多个 div)用绝对坐标定位(position:absolute)，使得多个层(div)都取相同的坐标，多张图片也就重叠了起来。初始状态下各层的 z-index 均设为 8。

另外还定义了一个类选择器，为后面的样式操作作准备。在这个类中只定义了深度坐

标 z-index:10。当某个层使用了这个类选择器时，代表该层处于最上层的位置。

(3) jQuery 代码。

```
<script type="text/javascript" src="Scripts/jquery.min.js"></script>
    <script type="text/javascript">
      function slideSwitch() {
       var $current = $("#slideshow div.current");
        // 判断 div.current 存在时则匹配$current.next(),否则转到第一个 div
       var $next = $current.next().length ? $current.next() :
                   $("#slideshow div:first");
       $current.removeClass("current");
       $next.addClass("current");
     };
      // 起始语句
      $(function() {
        setInterval("slideSwitch()", 4000);
       });
  </script>
```

代码分析如下。

● 下列语句是一个三元运算符。

```
var $next = $current.next().length ? $current.next() : $("#slideshow
             div:first");
```

代表当遍历到最后一个元素时(遍历情况参见图6.12)：

```
var $next = $("#slideshow div:first");
```

否则

```
var $next = $current.next();
```

● 下面是样式操作的语句。通过增、删“类”选择器，改变图片的深度。

```
$current.removeClass("current "); //取消前一个元素的 current 类选择器
$next.addClass("current");          //为被选中的元素增添 current 类(z-
index:10)
```

● 下面是定时调用前面函数的语句。

```
//起始语句
      $(function() {
        //每隔 4 秒(1000=1 秒) 执行一次上面的函数
        setInterval("slideSwitch()", 4000);
       });
```

(4) 用淡入淡出显示图片。

如果能让每次图片淡入淡出地显示，将会变得更加生动。要实现这一点只需利用 jQuery 中的 fadeIn()和 fadeOut ()方法即可。修改后的语句如下。

```
<style type="text/css">
  #slideshow{position:relative;height:50px;width:70px;margin:0 auto 5px ;}
  #slideshow div{position:absolute;top:0;left:0;z-index:8; width:
  70px;height:50px;
            border:2px solid #0000ff;overflow:hidden;background-
            color:#FFF;
```

```
                display:none;
                }
      #slideshow div.current{z-index:10;}

       #slideshow div img{
                      position:absolute;top:0px;left:0px;
                      display:block;border:0;margin-bottom:0px;}
    </style>
    <script type="text/javascript" src="Scripts/jquery.min.js"></script>
    <script type="text/javascript">
        function slideSwitch() {
          var $current = $("#slideshow div.current");
          $current.fadeOut(3000).removeClass("current");
          // 判断 div.current 存在时则匹配$current.next(),否则转到第一个 div
          var $next = $current.next().length ? $current.next() :
  $("#slideshow div:first");
          $next.fadeIn(3000).addClass("current");
        };

        $(function() {
         //  每隔 4 秒(1000=1 秒) 调用一次上面的函数
          setInterval("slideSwitch()", 4000);
      });
  </script>
  </head>
```

本代码与上面代码的主要区别是：

- 本代码中增加了改变不透明度的函数调用(fadeOut()与 fadeIn())；
- 在#slideshow div{…}的 CSS 定义中增加了 display:none 属性，使各图片在初始状态时都不显示。

6.8 插件(Plug in)

6.8.1 概述

1. 插件及其作用

一个 jQuery 插件(Plugin)实际上就是一段用 jQuery 编写的程序，用来提供给其他应用程序扩展功能。为了使插件能够通用，插件的接口必须遵循一定的规范。

使用插件就是为了实现软件重用的目标。一个 Web 设计者很容易利用插件来扩展功能，这是 jQuery 最成功的特点，这个特点不仅吸引了大批的 Web 设计者，同时也吸引了一大批顶尖的 JavaScript 专家来共同开发 jQuery 插件。到目前为止，已经开发出上千个优秀的插件，分别用于各种不同的应用程序中。

插件的官方网址是 http://plugins.jQuery.com/，该网站不仅包括插件的下载代码，也包括详细的使用说明，允许各设计者免费下载使用。

jQuery 插件按照功能可划分为表单验证插件(Form Validation)、管理 cookie 插件(Cookie)、ajax 应用插件(ajaxForm)、界面设计插件(jQuery UI)等。本章将重点讲述界面设计插件的使用。

2. jQuery UI 简介

jQuery UI 是 jQuery 插件的重要组成部分，用来实现 Web 中客户与浏览器的交互应用。它最初来源于一个 Interface 插件，后来在 Paul Bakaus 等人领导下，将 Interface 的大部分代码进行了重构，并统一了 API。由于改动很大，版本不是 1.3 而是直接跳到了 1.5，并改名为 jQuery UI，目前 jQuery UI 的最新版本是 1.8.7。随着 jQuery 本身内核的逐步完善，jQuery UI 很可能代表 jQuery 今后发展的走向。

jQuery UI 的官方网站是 http://ui.jQuery.com/download/。

2011 年年初提供了两种版本：Latest(1.8.7 for jQuery 1.3.2+)以及原有的稳定版本 Stable(1.7.3 for jQuery 1.3.2)。

按照功能划分，jQuery UI 插件可分为以下 4 部分。

- 核心库(UI Core)：包括一些基本函数和初始化函数。
- 交互(Interactions)：包括拖动(Draggable)、置放(Droppable)、缩放(Resizable)、选择(Selectable)、排序(Sortable)等。微件中部分插件是基于交互组件来制作的。交互库需要 jQuery UI 核心库——ui.core.js 的支持。
- 微件(Widgets)：主要是对一些界面的扩展。包括手风琴导航(Accordion)、自动完成(AutoComplete)、取色器(ColorPicker)、对话框(Dialog)、滑块(Slider)、标记(Tabs)、日历(DatePicker)、放大镜(Magnifier)、进度条(ProgressBar)等。此库需要 jQuery UI 核心库——ui.core.js 的支持。
- 效果库(Effects)：此库用于提供丰富的动画效果，这些动画不再局限于 animate()方法。效果库有自己的一套核心，即 effects.core.js，不需要 jQuery UI 核心库——ui.core.js 的支持。

6.8.2　简单插件的编写及使用

1. 编写一个最简单的插件

下面用一个简单的示例来说明撰写插件的一般方法，此插件的作用是为元素设置字符颜色。其代码如下。

```
;(function($) {
        $.fn.textcolor=
           function(bg) {
                return this.css("color",bg);
              }
          })(jQuery);
```

每个插件必须有一个名字，通常的形式是“$.fn.[插件名]”(如此例中的 textcolor)，使用者就是通过插件名来调用插件。插件的内容用 jQuery 语句编写。

有的插件前面先写一个分号(;)。为什么这么做？因为按照语法规定，jQuery 的语句之间要用分号分隔，而插件是被嵌入到其他程序段中的，如果被插入的插件前面的字句遗忘了分号，将带来不可预计的后果。而多加一个分号不会引起什么负作用。

另外，整个插件还用以下方式封装起来：

```
(function($){
```

```
        // 插件的定义
        })(jQuery);
```

为什么要封装？在 jQuery 语言中，用$符号代表 jQuery。而在插件前后的文件中，$可能还有其他含意，为了避免冲突，使用封装方式来说明，用$代表 jQuery 仅限于在此范围内有效。

最后插件要以.js(JavaScript 文件)作为后缀的文件保存。假定本例的插件存入的文件名为 cssrain.js。

2. 使用插件的方法

在网页中使用插件前，需要下载插件库，以及相关的 CSS 文件，并在网页中引用该库以及 CSS 文件。然后按照插件的要求配置 HTML 元素。最后编写 jQuery 代码通过插件名字来调用插件。例如：

```
 <html xmlns="http://www.w3.org/1999/xhtml">
<head>
 <title> 最简单的 Plugin </title>
  <script type="text/javascript" src="scripts/jquery-
                1.3.2.min.js"></script>
  <script type="text/javascript" src="scripts/cssrain.js"></script>
  <script type = "text/javascript">
   $(function(){
    $("table").textcolor("red");
    });
  </script>
 </head>
 <body>
  <table border="1" width="100px">
  <tr><td>1</td><td>1</td><td>1</td></tr>
  <tr><td>2</td><td>2</td><td>2</td></tr>
  <tr><td>3</td><td>3</td><td>3</td></tr>
  </table>
 </body>
 </html>
```

结果是表格中的字符均呈红色。

3. 当插件中包括多个参数时

有时插件中包括有多个参数以便引用者进行选择。此时最好将这些参数集中放置在一个地方。例如:

```
;(function($) {
$.fn.插件名 = function(o) {
    o = $.extend({
             参数名 1: 值 1,
             参数名 2:值 2,
             …
            }, o || {});
  })(jQuery);
```

6.8.3 使用 jQuery UI 插件的范例

jQuery UI 的插件很多，使用时应先看清该插件的使用说明，以避免走弯路。下面通

过两个范例来说明插件的使用方法。

例 6.3　使用 jCarouselLite 插件。

本插件的显示界面如图 6.13 所示。

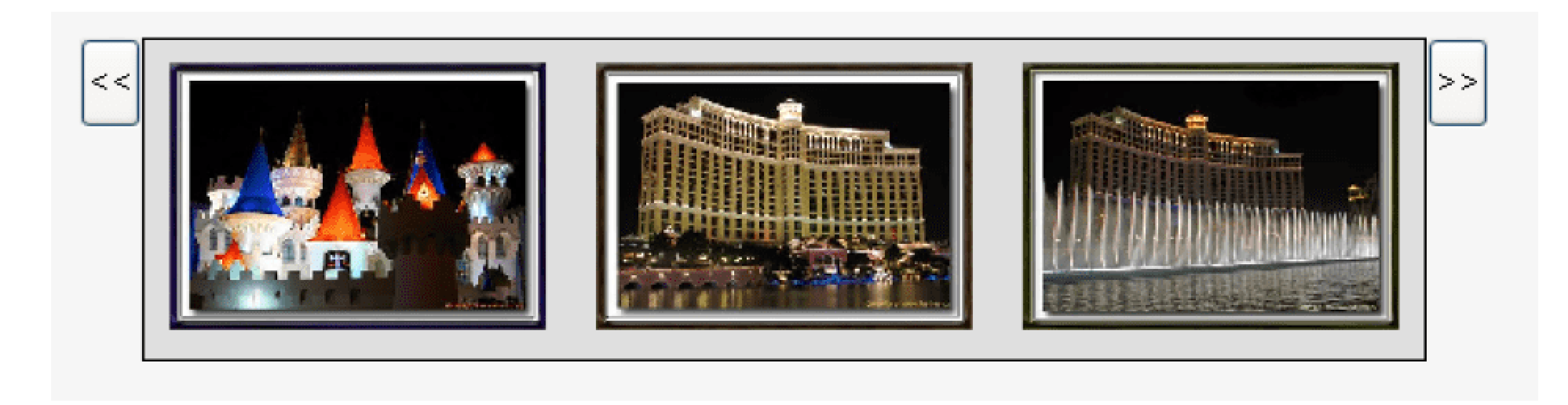


图 6.13　jCarouselLite 插件的显示界面

本插件的功能是滚动地浏览多张图片。滚动的方向既可以是水平方向也可以是垂直方向，可以定时滚动也可以用按钮控制滚动。这个插件的最大特点是可以将图片的首尾相接，实现无缝地循环显示。(注：早期的版本不具备无缝循环的功能)

调用的步骤如下。

(1)　下载插件。

插件下载的 URL 是 http://gmarwaha.com/jQuery/jcarousellite/，并在网页中指向插件。例如：

```
    <script type="text/javascript" src="Script/jQuery-
latest.pack.js"></script>
<script type="text/javascript"
src="Script/jcarousellite_1.0.1.js"></script>
```

注意：注意上面使用的版本，因为不同版本运行的结果可能会有稍许区别。

(2)　按照 jcarousellite_1.0.1.js 文件的要求对图片进行布局。

布局的结果如下。

```
<div class="carousel">
       <ul>
           <li><img src="images/图片小1.JPG" alt="1"></li>
           <li><img src="images/图片小2_JPG" alt="2"></li>
           <li><img src="images/图片小3.JPG" alt="3"></li>
           …
       </ul>
   </div>
```

(3)　按要求设置按钮。

```
<button class="prev">&lt;&lt;</button>
<button class="next">&gt;&gt;</button>
```

(4)　修改参数。

插件提供了多种参数，这些参数的设置方法均可从 jcarousellite_1.0.1.js 文件中看到。下面举几个比较常用的参数加以说明。

- Auto：默认为 null，表示不能自动进行循环。若设为 800 代表自动循环，转换的间隔为 800 毫秒。
- Speed：默认为 200 毫秒，这代表图片逐步显示出来的速度。
- Verticle：默认为 false。表示不按垂直方向显示(按水平方向显示)。
- Visible：默认为 3，表示每次显示 3 张图片，改成 3.5 或 4 时表示每次显示 3.5 或 4 张图片。

(5) 编写代码调用插件。

代码如下。

```
<script type="text/javascript">
      $(function() {
         $(".carousel").jCarouselLite({
            btnNext: ".next",
            btnPrev: ".prev"
         });
      });
   </script>
```

例 6.4 使用 JQZoom 插件。

本插件用来模拟放大镜的功能。在电子商务的商品展示中，如果允许顾客用此插件来放大商品的某些局部，可以让客户进一步找到自己满意的商品。本插件的使用步骤如下。

(1) 下载相关文件。下载的文件包括插件文件(.js)、CSS 文件(.css)以及多张成对的大小图片(.jpg)。

① 下载插件然后用下列代码指向它。相关代码如下。

```
<script type="text/javascript" src="script/jQuery.js"></script>
<script type="text/javascript" src="script/jQuery.jqzoom.js"></script>
```

jQuery jqzoom.js 中各参数的含义在使用文件中有说明。主要有以下几项。

- zoomType 'standard'：'reverse'默认'standard'，'reverse'的时候开启透明。
- imageOpacity：默认 0.2 透明度开启'reverse'的时候可用。
- zoomWidth：宽度 zoomHeight 高度 默认都是 200。
- xOffset：默认 10 x 方向放大效果 div 的偏移。
- yOffset：默认 0 y 方向放大效果 div 的偏移。
- position：默认'right'可选：'right'，'left'，'top'，'bottom'放大效果的位置。
- lens：默认 true 被放大区域是否突出。
- title：默认 true 标题。
- showEffect：默认'show'可选'show'，'fadein'开启渐入效果。
- hideEffect：默认'hide'可选'hide'，'fadeout'开启渐出效果。
- fadeinSpeed fadeoutSpeed：可选'fast'，'slow'，'medium'默认'slow'出入的速度。
- showPreload true/false：是否显示加载。
- preloadText：默认 Loading zoom 加载时的标题。
- preloadPosition：默认 center 加载区位置(可自行更改 css)。

② 下载 jqzoom.css 文件并用以下代码指向该文件。

```
<link rel="Stylesheet" type="text/css" href="css/jqzoom.css" />
```

上述样式中包括对 class="jqzoom"的定义和大图的属性"jqimg"。

③　下载大、小对应的两组图片，图片根据需要自行选择。

(2)　编写 HTML 代码。

本例中编写的代码如下。

```
<body>
 <form id="form1" runat="server">
    <div>
    <p class="locat1">
     <div class="jqzoom">
       <img alt="1" src="images/图片小 1.JPG" jqimg="images/图片 1.JPG"
       /></div>
     <div class="jqzoom">
       <img alt="2" src="images/图片小 2.JPG" jqimg="images/图片 2.JPG"
       /></div>
     <div class="jqzoom">
       <img alt="3" src="images/图片小 3.JPG" jqimg="images/图片 3.JPG"
       /></div>
     <div class="jqzoom">
       <img alt="4" src="images/图片小 4.JPG" jqimg="images/图片 4.JPG"
       /></div>
     <div class="jqzoom">
       <img alt="5" src="images/图片小 5.JPG" jqimg="images/图片 5.JPG"
       /></div>
     <div class="jqzoom">
       <img alt="6" src="images/图片小 6.JPG" jqimg="images/图片 6.JPG"
       /></div>
    </p>
   </div>
   </form>
</body>
```

另外还需编写一个 CSS 定义。

```
<style type="text/css">
     .locat1{
        width:863px;
        height:200px;
        position:relative;
        top: 212px;
        left: 30px;
        }
</style>
```

(3)　编写调用插件的代码。

调用插件的代码如下。

```
<script type="text/javascript">
        $(function() {
            $(".jqzoom").jqueryzoom();
        });
```

```
</script>
```

也可以在调用的同时改变大图的显示参数，具体修改的方法参见该插件的使用文件。配置好以后运行的结果如图 6.14 所示。



图 6.14 使用 jQZoom 插件的情况

6.9 小 结

jQuery 可以无缝地融入到 ASP.NET 应用程序之中。它的主要任务就是对浏览器的 DOM 元素进行操作。它是一种优秀的 JavaScript 语言库，其目的在于简化对 JavaScript 语言的使用，利用它能够用简短的语句完成较复杂的工作。

jQuery 采用了与 CSS 类似的定位方法，即用 jQuery 定位的是一组对象集而不是单个的 DOM 对象。jQuery 中的方法只能用于 jQuery 的集合对象。当然这种集合对象与 DOM 单个对象之间的转换也非常容易。

jQuery 库的容量很小，语句非常简练，jQuery 的方法语句可以连写，这些特点使它非常适合于网络的应用，这是它的重要特点。

jQuery 提供了大量插件，供广大设计者无偿使用，而且这些插件随着 Web 的发展不断增加，全世界几乎所有用户社区(包括 Microsoft 社区)和对此感兴趣的开发人员都在不断地改进它，这是系统最吸引人的地方。然而要想掌握 jQuery 技术必须学会它的基本概念，特别要搞清在 DOM 基础上将 HTML 标记、CSS 以及 jQuery 语句三者联系起来设计动画的方法。

6.10 习 题

1. 填空题

(1) jQuery 主要应用于____________、____________、____________三个领域。

(2) jQuery 的起始语句是为了保证所有的____________元素下载完毕后才执行后续语句。

(3) 用$("div .content")定位了____________元素。

(4)　语句$("p").addClass("myclass");代表______________。

(5)　语句$("p").removeClass("anotherclass");代表______________。

(6)　在某插件的前面出现了如下代码。

```
;(function($){
    $.fn.textexp = function(bg){…}
        })(jQuery);
```

引用该插件的名字是____。

2. 选择题

(1)　在 jQuery 对 DOM 的遍历中，用 children()方法将获得______________。

A. 本元素儿子的相对路径

B. 本元素儿子的元素集

C. 本元素儿子的个数

D. 本元素儿子的绝对路径

(2)　在 jQuery 对 DOM 的遍历中，用 siblings()方法将获得______________。

A. 本元素的相对路径

B. 本元素的兄弟元素集

C. 本元素儿子的个数

D. 本元素的绝对路径

(3)　在 jQuery 中$(this)是一个关键字，它代表______________。

A. 正在运行的 jQuery 对象

B. 匿名的 DOM 对象

C. 当前被鼠标选中的 DOM 对象

D. DOM 遍历中的下一个对象

3. 判断题

(1)　jQuery 是与 JavaScript 语言无关的语言。　(　　)

(2)　jQuery 语言的主要任务就是对 DOM 元素进行操作。　(　　)

(3)　jQuery 定位的方式与 CSS 定位的方式基本相同。　(　　)

(4)　jQuery 语言中的方法只能用于 jQuery 对象。　(　　)

(5)　jQuery 元素就是 DOM 的单个元素。　(　　)

4. 简答题

(1)　在某个插件的前面出现了如下代码。

```
;(function($){
    $.fn.textexp = function(bg){…}
        })(jQuery);
```

讲述代码中两处带下划线的符号的作用。

(2)　说明下列 jQuery 代码段的作用。

```
$(".square").animate({opcity : "0.1" ,left : "+=50"}, 3000)…
```

5. 操作题

(1) 习题要求与综合示例 2 相同，但要求用上拉幕布方式显示图片。

(2) 从因特网上自选三个插件(Plugin)运行于自己的网页中。

第三部分

服务器端开发

ASP.NET 是一个基于服务器的系统，服务器端开发是网站开发的基础，也是本书讲授的重点。这一部分将用 12 章的篇幅讲述三方面的内容。

- ASPX 网页以及各种服务器端控件的使用方法。
- 数据访问技术。包括 ADO.NET 的系统结构，对数据库的连接、显示、编辑与同步，以及存储过程、数据缓存的方法等。
- LINQ 技术。

第 7 章　ASPX 网页及代码的存储模式

用 ASP.NET 3.5 创建的网站中虽然可以包括多种类型的网页，如.htm 文件、.asp 文件等。但是最基本的网页是以.aspx 作为后缀的网页，这种网页又称为 ASPX 网页(或称为 Web 窗体页)。从客户的角度看来，好像这些网页的使用并没有什么区别，但实际上，它们背后的运行机制却有本质的不同。从本章开始，将从网页的代码存储模式、事件模型以及状态管理三个方面来讲述 ASPX 网页的运行机制。本章中将要讲述的问题包括：

- ASPX 网页的基类。
- ASPX 网页代码的存储模式。
- ASPX 网页中的表单。

7.1　ASPX 网页的基类

ASP.NET 是一个完全面向对象的系统，网页是网站的基本组成部分。每张 ASPX 网页都直接或间接从类库中的 System.Web.UI.Page 类继承。由于在 Page 类中已经定义了网页所需要的基本属性、事件和方法。因此只要新网页生成，就从它的基类中继承了这些成员，因而也就具备了网页的基本功能。设计者可以在这个基础上高起点地进行各项设计。

例如，在 Page 类中已经定义了以下成员。

- Request：这是一个与 HTTP 通信的属性，该属性返回 HTTP Runtime Request 对象。通过这个对象可以获取来自 HTTP 请求的数据。
- Response：这也是一个与 HTTP 通信的属性，返回 HTTP Runtime Response 对象。和 Request 对象的作用正好相反，这个对象允许向浏览器端发送信息。
- ViewState、Session、Application 对象：这些对象用来保持网页的各种状态。具体内容在后面章节中讲述。
- Init、Load 等事件：网页初始化和刚被装载时触发的事件。

7.2　ASPX 网页代码的存储模式

每个 ASPX 网页中实际上包含两方面的代码：用于显示的代码和用于逻辑处理的代码。用于显示的代码包括 HTML 标记以及对 Web 控件的定义等；用于逻辑处理的代码主要是用 C#.NET(或其他语言)编写的事件处理程序。

在 ASPX 网页中，这些代码可以用两种模式存储：一种是代码分离模式，另一种是单一模式。在代码分离模式中，显示信息的代码与逻辑处理代码分别放在不同的文件中；在单一模式中，将两种代码放置在同一个文件中。

创建 ASPX 网页时就可以选择存储的方式。设置的方法如图 7.1 所示。

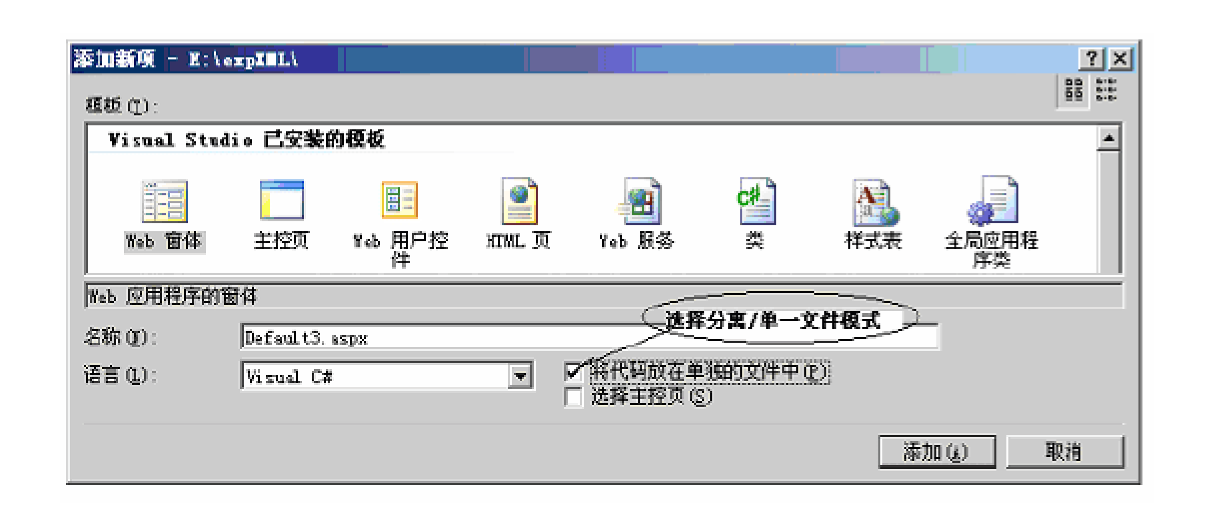


图 7.1　选择代码存储模式

对话框右下角的第一个复选框用来确定存储模式。如果被选中，将使用分离模式，否则使用单一模式。

7.2.1　代码分离模式

在代码分离模式中用于显示的代码(HTML 标记、服务器控件的定义等)将仍然放在后缀为.aspx 的文件中，而用于逻辑处理的代码放到另一个文件中，该文件的后缀依据使用的程序语言而确定。如果使用 C#.NET 语言时，文件的后缀是.aspx.cs；如果使用 VB.NET 语言时，文件的后缀是.aspx.vb。此文件有时又可称为代码隐藏(Code-Behind)文件。

1. 逻辑处理代码文件

逻辑处理代码文件是一个类文件。该文件示例如图 7.2 所示。

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        |
    }
}
```

图 7.2　逻辑处理的代码框架

这是一个 Default.aspx 网页中的逻辑处理代码文件 Default.aspx.cs。当网页刚被创建时虽然还没有编写任何代码，但系统已经给出了网页代码的初步框架。

1)　定义类的基类

下面的语句是对网页类定义的框架：

```
public partial class _Default: System.Web.UI.Page
  {
    …
   }
```

表明网页是一个类，派生于 System.Web.UI.Page 基类。在类的定义中修饰词 partial class 代替了传统的 class。这说明网页是一个“分布式类”。

所谓分布式类是 C#.NET 2.0 中新增加的一种数据类型。

有的“类”具备比较复杂的功能，因而拥有大量的字段、属性、事件和方法，甚至还可能包括大量的嵌套成员。如果将“类”的定义都写在一起，文件一定庞大，代码的行数一定很多，不便于理解和调试。为了降低文件的复杂性，C#.NET 语言提供了“分布式类”的概念。

在分布式类中，允许将“类”的定义分散到多个代码片段之中，而这些代码片段又可以存放到两个或两个以上的源文件中，每个文件只包括类定义的一部分。只要各文件中使用了相同的命名空间、相同的类名，而且每个类的定义前面都加上 partial 修饰符，编译时编译器就会自动将这些文件编译到一起，形成一个完整的类。

例如：

```
// 第一个文件名为 exp1.cs
using System;

public partial class partexp
{
public void SomeMethod()
{
}
}

// 第二个文件名为 exp2.cs
using System;

public partial class partexp
{
public void SomeOtherMethod()
{
}
}
```

上面 exp1.cs 与 exp2.cs 两个文件都使用同一命名空间(System)、同一类名(partexp)，而且都加上了 partial 修饰符。编译后生成的类将自动将两个方法组合到一起。结果新类包括了两个方法：SomeMethod()和 SomeOtherMethod()。

和 ASP.NET 1.x 版本不同，由于采用分布式类的结构，代码文件已被大大简化，定义中只包括新增加的代码部分。此时的文件结构如图 7.3 所示。

2) 命名空间的引用

文件前面包含一系列命名空间的引用。例如：

```
using System;
using System.Data;
using System.Web;
…
```

这些命名空间都是网站中经常需要用到的部分。默认情况下自动显示，不需要手动编写。

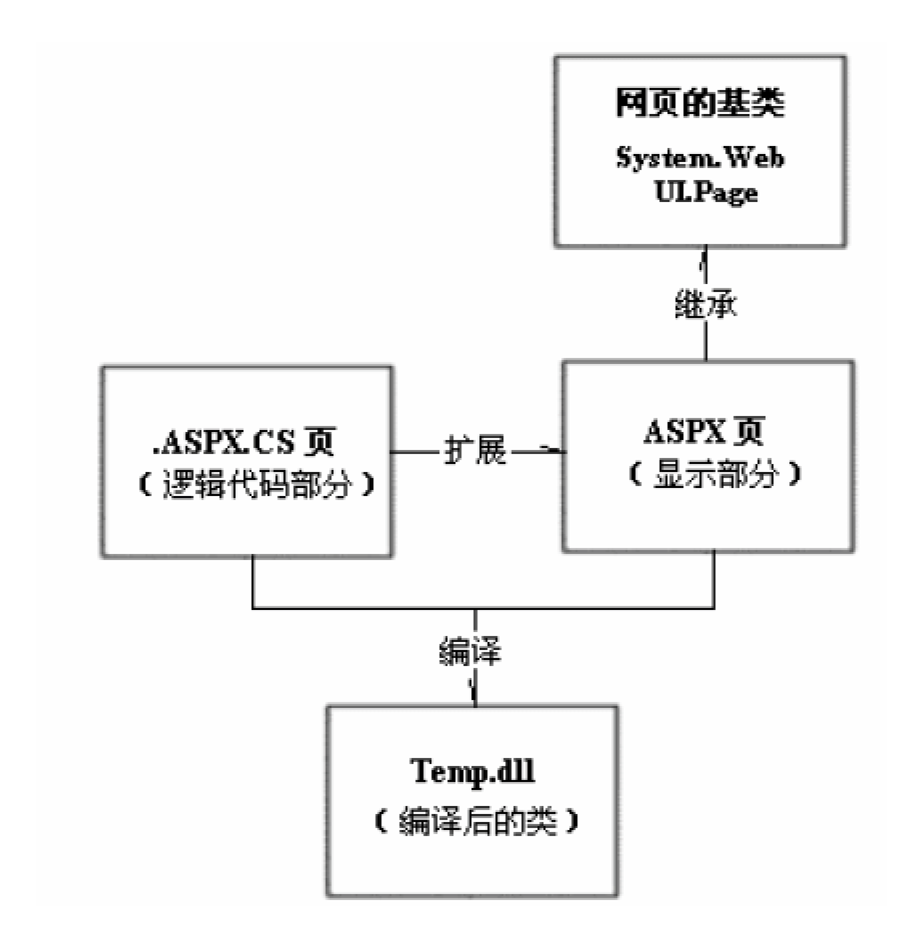


图 7.3　分布式类的结构

3)　事件处理代码

事件处理代码作为类的成员包括在类的定义中。初始情况下系统只给出了网页被装载(Page_Load)事件的代码框架，只要网页被调用，一打开就执行本事件中的代码。例如：

```
protected void Page_Load (object sender,  EventArgs e)
{
…
}
```

2. 显示代码文件(Defualt.aspx)

打开 Default.aspx 网页，代码如图 7.4 所示。

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
        Inherits="Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
        "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">
    <title>Untitled Page</title>
</head>

<body>
    <form id="form1" runat="server">

    </form>
</body>
</html>
```

图 7.4　用于显示网页的框架

代码包括对若干网页的定义。前 3 行指令对网页进行了定义。如定义了使用的语言(Language="C#")、逻辑处理代码文件的名称(CodeFile="Default.aspx.cs")等。

代码中的

```
<form id="form1" runat="server">
…
</form>
```

表明每个网页是一个表单，表单中能够放置各种表单控件，利用这些控件可以收集或显示各种信息，并在浏览器与 Web 服务器之间进行交互。

7.2.2 代码的单文件模式

在代码的单文件模式中，用于显示的代码与逻辑处理代码都放在同一个后缀为.aspx 的文件中。文件中的逻辑处理代码(事件、方法或属性)放在用<script>…</script>标记包括的模块中，以便与其他显示代码隔离开。需要在服务器端运行的代码一律在<script>标记中注明 runat="server"属性。一个模块可以包括多个程序段，每个网页也可以包括多个<script>模块。

单文件模式的继承关系如图 7.5 所示。下面就是一个简单的网页单文件示例，设文件名为 exp4_1.aspx。文件的内容如下。

```
<%@ Page Language="C#" %>
  <script runat="server">
  void Button1_Click(Object sender, EventArgs e)
    {
     Label1.Text = "Clicked at " + DateTime.Now.ToString();
    }
   </script>

  <html>
    <head>
      <title>Single-File Page Model</title>
    </head>
    <body>
      <form runat="server">
        <div>
          <asp:Label id="Label1"
           runat="server">Label
         </asp:Label>
         <br />
         <asp:Button id="Button1"
         runat="server"
         onclick="Button1_Click"
         Text="Button"></asp:Button>
       </div>
      </form>
    </body>
</html>
```

这是一个后缀为.aspx 的网页文件，此网页中包含一个 Button 控件和一个 Label 控件。代码中的<script>…</script>模块中定义的是一段事件处理代码，当单击 Button 控件时将执行这段代码，即显示单击的日期和时间。

允许在同一个网页中使用不同类型的脚本，如图 7.6 所示。

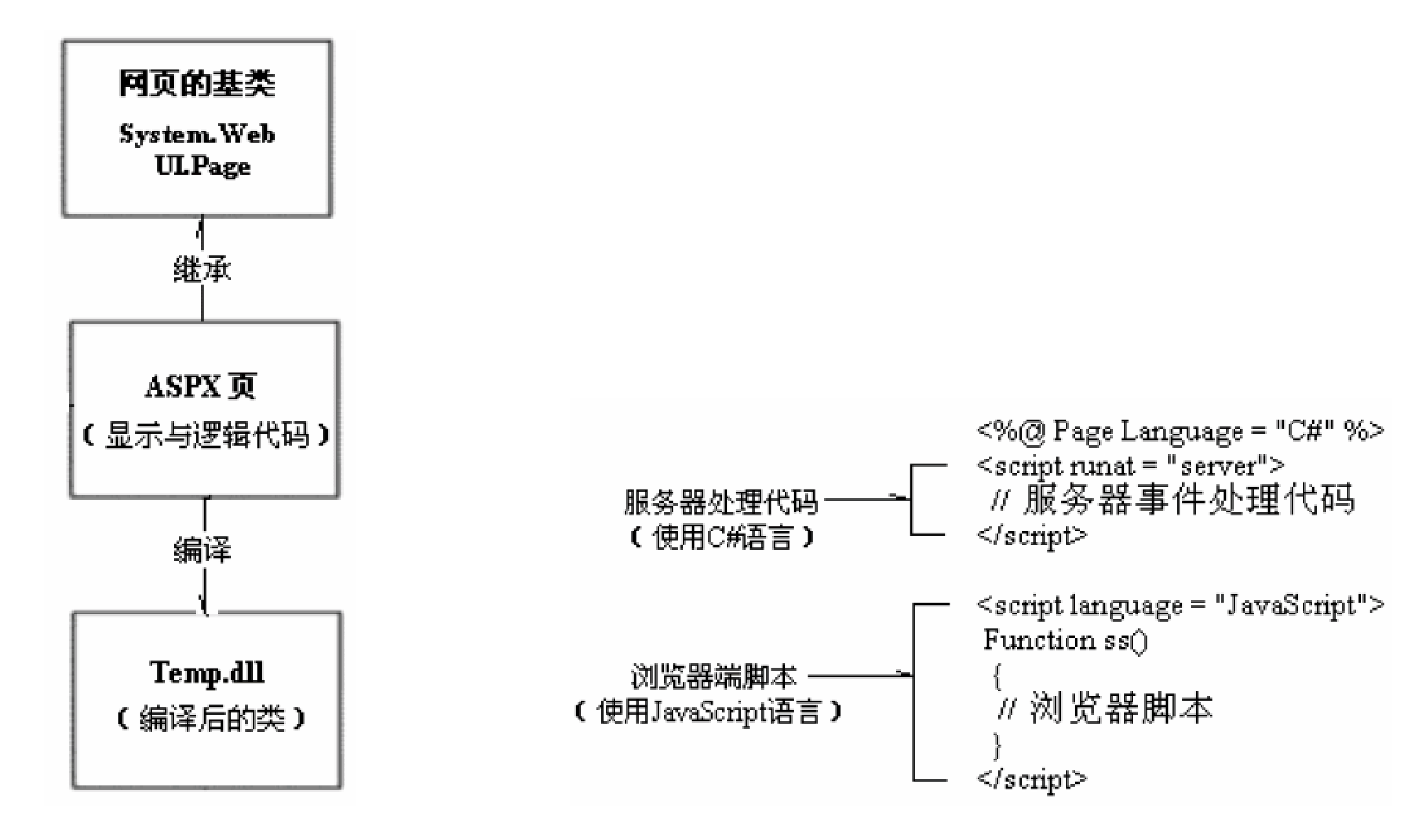


图 7.5　网页单文件的继承方式　　**图 7.6　同一网页中的两种脚本**

其中第一段是用 C# 语言编写的服务器事件处理代码；第二段是用 JavaScript 语言编写的浏览器端脚本。两者放在同一张网页中分别执行服务器处理的事件和浏览器端处理的事件。

7.3　ASPX 网页中的表单

ASPX 网页的 Web 表单虽然是从 HTML 表单发展而来的，但是两种表单的运行机制却有本质的区别。

ASP.NET 是基于服务器的事件驱动模型。与 HTML 表单相比在 Web 表单的定义中多一个 runat="server"属性，表明该表单也是基于服务器的。每个 ASPX 网页实际上就是一个 Web 表单，网页中的控件都直接包含与服务器进行交互的数据，各种按钮实际上都起到提交按钮的作用。

Web 表单的定义如下：

```
<form id="form1" runat="server">
…
</form>
```

Web 表单的交互过程如图 7.7 所示。

这个过程说明，由于服务器控件与服务器进行数据绑定，因此不论控件的生成，还是对事件的处理基本上都是在服务器端进行，而且处理完毕后仍然返回到本网页。因此在 HTML 表单中的 Action 和 Method 属性在 Web 表单中都不再需要了。

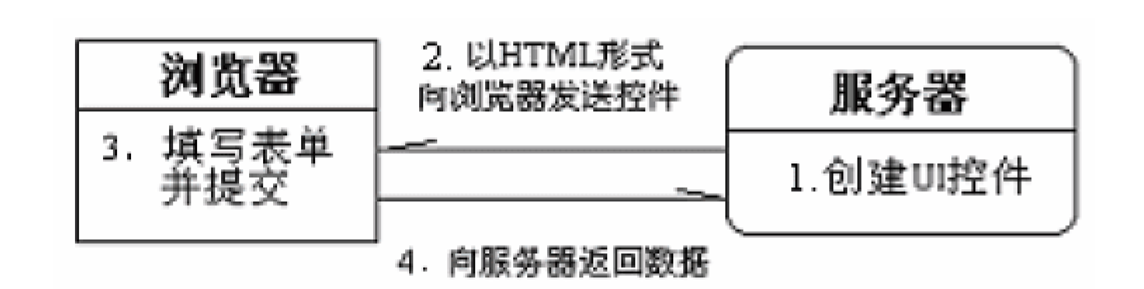


图 7.7 Web 表单的交互过程

7.4 代码模式的选择

ASP.NET 提供了单文件和代码分离两种模式。两种模式各有优点，概括地说：代码分离模式可以使设计思路变得更加清晰；单文件模式则由于能够统观全局，更有利于看清控件与代码之间的联系。

一般来说，对于那些逻辑代码不太复杂的网页，最好采用单文件模式；而对于逻辑代码比较复杂的网页来说，最好采用代码分离模式。

7.5 小 结

ASPX 网页是从 Page 类继承的类，因此当网页刚被创建时就已经具备有基本功能。网页的存储可以分为两种模式：单文件模式和代码分离模式。ASP.NET 2.0 以及后面版本对二者都进行了改进。虽然两种模式最后的功能相同，但是不同模式还是有自己的特点。对于逻辑代码比较复杂的网页来说最好采用分离模式，简单的网页适合采用单文件模式。每个 ASPX 网页实际上就是一个表单，是一个运行在服务器端的表单。在表单中放置的控件虽然显示在浏览器端，但基本上都可以与服务器进行交互，交互的过程参见第 8 章的事件模型。

7.6 习 题

1. 填空题

(1) ASPX 网页的基类是________________。

(2) ASPX 网页的代码存储模式有两种，它们是__________和__________。

(3) 所谓分布式类就是在多个文件中使用相同的________、相同的_____，而且每个类的定义前面都加上________修饰符，编译时编译器就会自动将这些文件编译成一个完整的类。

(4) 若使用 C#语言，在代码分离模式中逻辑代码的文件后缀是________。

2. 判断题

(1) 对于逻辑代码比较复杂的类来说最好采用代码分离模式。 ( )

(2) 代码分离模式的网页运行效率要高于单一模式。 ( )

3. 简答题

(1) 举例说明浏览器端脚本与服务器端脚本定义语句的区别。

(2) 简述代码存储的分离模式和单一模式各自的特点。

(3) ASPX 网页中的表单与.htm 网页中的表单有哪些区别？

第 8 章　标准控件与事件模型

控件是一种类，绝大多数控件都具有可视的界面，能够在程序运行中显示出其外观。利用控件进行可视化设计既直观又方便，可以实现“所见即所得”(What You See Is What You Get，WYSIWYG)的效果。程序设计的主要内容是选择和设置控件以及对控件的事件编写处理代码。

本章将介绍网页中最常用的几个标准网页控件，目的在于学会这些常用控件的使用方法；然后重点介绍 ASP.NET 3.5 的事件模型。本章的主要内容包括：

- 网页中的控件。
- ASP.NET 的事件处理模型。
- 应用示例。

8.1　网页中的控件

8.1.1　控件类型

ASP.NET 的类库中包括大量的控件，根据功能可以将它们分成以下几种类型。

- HTML 控件：属于客户端(浏览器端)控件。
- 网页标准控件：是服务器端常用控件。
- 数据控件：用于数据访问的控件。
- 验证控件：用于验证客户的输入。
- 导航控件：用于网站导航。
- 登录控件：用于客户登录。
- Web Parts 控件：用于个性化服务。
- Ajax 扩展控件：用于实现 Ajax 技术。
- 用户控件及自定义控件：这些控件都是由程序设计者自行定义的控件，是对系统控件的扩展。

8.1.2　网页标准控件

在 ASP.NET 3.5 的工具箱中，只有 HTML 选项卡中的控件属于浏览器端控件，其他各种控件都是服务器控件。其中【标准】选项卡中的控件是常用的控件。在类库中，所有的网页控件都是从 System.Web.UI.Control.WebControls 直接或间接派生而来的。

1. Web 标准控件的功能

在工具箱的 Standard 选项卡中包括有几十个标准控件。这些控件中既有传统的窗体控件，例如按钮、复选框、文本框等，还有用来显示数据、选择日期等比较复杂的控件。其中，有的控件还具有很高的智能，例如：

- 能自动检测浏览器的类型，并根据浏览器的类型提供不同的输出。
- 能够使用模板来定义控件的外观。
- 可以选择事件信息传送的方式，是立即发送给服务器，还是先缓存然后再和其他信息一起传送给服务器。
- 有的控件可将事件信息从嵌套控件(例如表中的按钮)传递到它的容器控件。

2. 定义标准控件的格式

定义标准控件的格式如下。

```
<asp:Control id="name" runat = "server" />
```

其中 asp 代表命名空间，所有的 Web 服务器控件的命名空间都是 asp；Control 代表控件的类型；id 代表控件的标志；runat="server"代表这是一个服务器控件，默认情况下，所有标准控件都是服务器控件。

例如，定义 Label、TextBox、Button 等网页控件的代码如下。

```
<asp:Label id="Label1" runat="server"/>
<asp:TextBox id="TextBox1" runat="server"/>
<asp:Button id="Button1" runat="server"/>
```

3. 标准控件作用的列表

标准的服务器控件的作用比较广泛。表 8.1 归类列出了部分服务器控件及作用。

表 8.1　部分服务器控件及作用

作　用	控件类型	说　明
文本显示(只读)	Label	只能用于显示，不能直接编辑的文本
文本输入	TextBox	用于输入、显示并可以编辑的文本框
下拉列表	DropDownList	可以输入或供选择的下拉列表
	ListBox	显示可选多列的列表
图形显示	Image	显示图像
	AdRotator	用于显示一系列图像
复选框	CheckBox	用于多项选择的复选框
单选按钮	RadioButton	是、否选择按钮
设置日期	Calendar	用于显示并选择日期
按钮	Button	执行某项任务
	LinkButton	执行某项任务，但外观类似超链接
	ImageButton	执行某项任务，但外观显示为图片
导航控制	HyperLink	创建网页超链接
表格	Table	创建表格
	TableCell	在表的某行中创建一个单独的单元格
	TableRow	在表中创建一行

续表

作 用	控件类型	说 明
分组选择	CheckBoxList	创建复选框的集合
	RadioButtonList	创建单选按钮的集合
面板	Panel	建立一个无边界的区间，作为其他控件的容器
列表控制	Repeater	重复显示数据集中的记录
	DataList	和 Repeater 类似，但具有更多的格式和布局的选择
	DataGrid	带有行、列的数据表，有较强的编辑功能
占位	PlaceHolder	相当于一个空容器，可动态添加元素
转换	Literal	将静态文本转换为网页，不添加 HTML 元素
读入 XML	XML	将 XML 写入 Web 窗体控件中

1) Label 控件

Label(标签)控件是最简单的控件，该控件的功能就是输出文本，该文本可以通过它的属性 Text 直接设定，也可以利用程序动态地设定。定义的语法如下。

```
<asp:label id = "Label1" runat = "server">输出文本</asp:label>
```

或

```
<asp:label id = "Label1" runat = "server" Text="输出文本" />
```

2) TextBox 控件

TextBox(文本框)控件是用得最多的控件之一，该控件可以用来显示数据或者输入数据。定义的格式如下。

```
<asp:Textbox id="TextBox1" Text = "请在这里输入数据"Column="25"
MaxLengh="35" runat="server" />
```

TextBox 控件有一个重要的属性：TextMode。该属性包括 3 个选项。

- SingleLine：单行编辑框。
- MultiLine：带滚动条的多行文本框。
- PassWord：密码输入框，所有输入字符都用特殊字符(例如“*”)来显示。

3) Button、LinkButton 和 ImageButton 控件

Web 控件中的按钮分为三种：Button、LinkButton 和 ImageButton。它们之间功能相同，只有外观上有区别。Button 的外观与传统按钮的外观相同；LinkButton 的外观与超链接字符串相同；ImageButton 按钮用图形方式显示外观，其图像通过 ImageURL 属性来设置。

三种按钮的功能都与 HTML 的提交按钮相似，即每当这些按钮被单击(Click)时，就将缓冲区中的事件信息一并提交给服务器。

定义按钮的语法如下。

```
<asp:Button id="Button1" runat="server"Text="按钮"></asp:Button>
<asp:LINKbutton id="LinkButton1"runat="server">链接按钮</asp:LinkButton>
<asp:ImageButton
```

```
id="ImageButton1"runat="servervImageUrl="…"></asp;ImageButton>
```

双击 LinkButton 按钮，然后在隐藏文件中给按钮的 Click 事件写出以下程序，该按钮即可通过服务器转向新的网页，从而起到超链接的作用。

```
private void LinkButton1_Click(object sender, System.EventArgs e)
{
    Response.Redirect("其他窗体的 URL");
}
```

另外，3 种按钮都有一个 PostBackUrl 属性，利用这个属性可以将按钮变成返回按钮。即先将该属性设置成某个网页的 URL，以后单击该按钮时就会直接转向该网页。

4)　CheckBox 及 CheckBoxList 控件

CheckBox 和 CheckBoxList(复选)控件为客户提供了在真/假、是/否或开/关选项之间进行选择的方法。CheckBox 是单个控件，可以单独使用。而 CheckBoxList 控件是复选框列表控件，可以将多个 CheckBox 组合在一起使用。使用单个 CheckBox 控件时，更容易控制页面上的布局。例如，可以在各个复选框之间包含文本，也可以单独控制复选框的字体和颜色。如果想用数据库中的数据创建一系列复选框，则 CheckBoxList 控件是较好的选择。

使用 CheckBoxList 时要给控件增添选项。方法是先选择该控件，然后找到控件的 Items 属性，单击右边的省略号按钮，将弹出如图 8.1 所示的对话框。

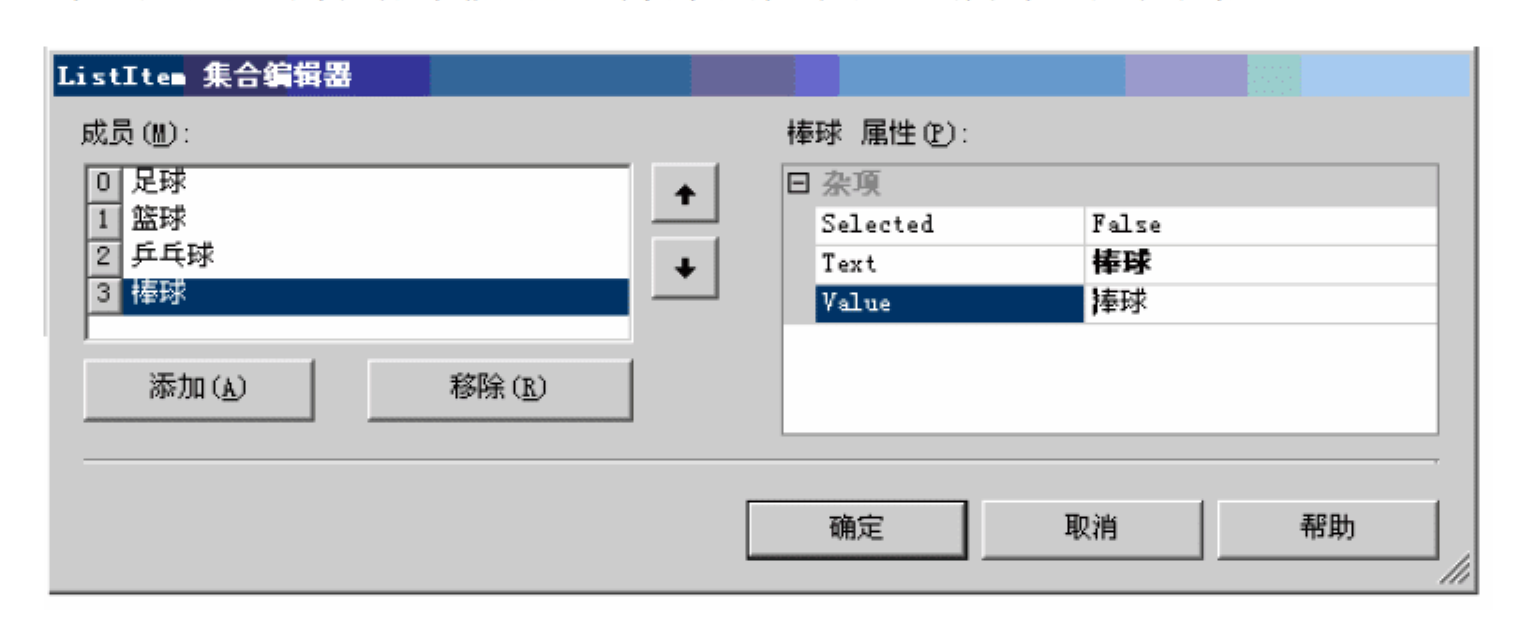


图 8.1　对复选框 Items 属性的设置

利用【添加】按钮逐个增加选项。按照图 8.1 中的选择得出的结果如下。



HTML 中的 CheckBox 控件只能用于静态处理，而网页控件中的 CheckBox 可以进行数据绑定操作，所以通常用于动态数据绑定。关于数据绑定，将在后面章节中介绍。

5)　RadioButton 和 RadioButtonList 控件

RadioButton 和 RadioButtonList(单选)控件的作用和使用方法与 CheckBox 基本相同，唯一的差别在于，在一个 RadioButtonList 内的多个 RadioButton 之间只能有一项被选中，而在 CheckBoxList 中可以同时选中多项。

6) Image 与 ImageMap 控件

利用 Image(图像)控件可以在 Web 窗体页上显示图像，并用自己的代码来管理这些图像。图像源文件可以在设计时确定，也可以在程序运行中指定，还可以将控件的 ImageURL 属性绑定到数据源上，根据数据源的信息来选择图像。

与大多数其他 Web 服务器控件不同，Image 控件不支持鼠标单击事件。如果需要使用鼠标单击事件时，可以使用 ImageButton 控件来代替 Image 控件。

显示一个图像所需的最少操作是：先创建一个 Image 控件，然后指定一个图像文件。具体步骤如下。

(1) 打开【设计】视图，在工具箱中打开【标准】选项卡，然后将一个 Image 控件拖放到网页界面上。

(2) 将控件的 ImageURL 属性设置为.gif、.jpg 或其他网络图形文件的 URL。

(3) 给 Image 控件设置以下属性。

- Height 和 Width：在页面上为图形保留适当空间(高度和宽度)。
- ImageAlign：用来设置图像对齐的方式。可使用的值包括 Top、Bottom、Left、Middle 和 Right。
- Alternatetext：有的浏览器不支持加载图像时，替代图像的文本。

ImageMap 控件可以用来显示图像，也可以实现图像的超链接。该控件的最大特点是，可以将 ImageMap 中的图像按照(x,y)坐标划分成不同形状的区域，分别链接到不同的网页。该控件的 ImageUrl 属性用来连接图像源文件；HotSpot 属性用来划分链接区域。单击 HotSpot 属性右边的省略号按钮，将弹出如图 8.2 所示的对话框。

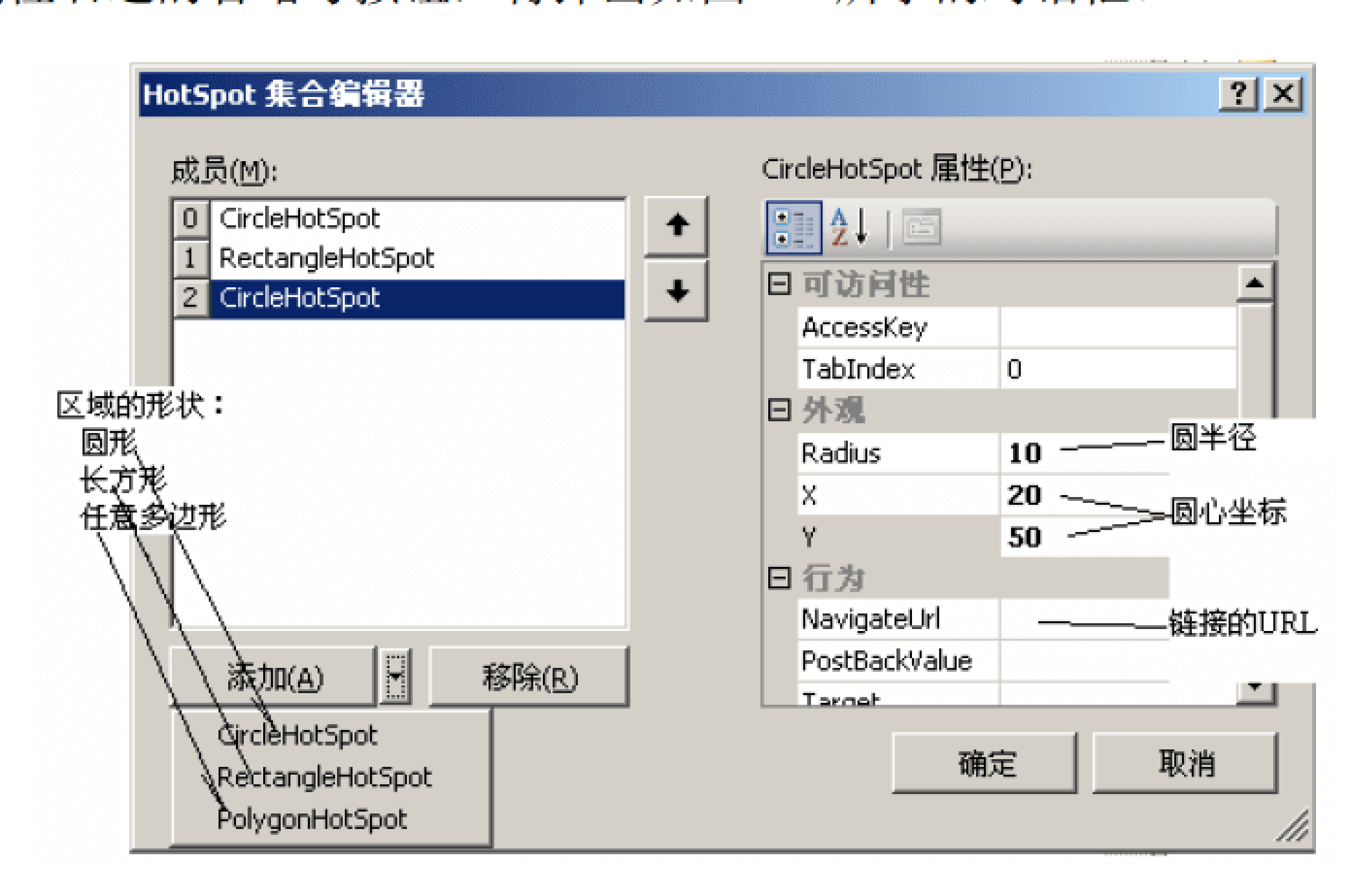


图 8.2 划分链接区域

利用【添加】按钮的下拉列表可以选择区域的形状，在右边的属性中可以确定区域的位置以及链接的网页。

7) HyperLink 控件

HyperLink(超链接)控件用于从一个页面转向到另一个页面，与 HTML 中的超链接不同之处在于，此处的 HyperLink 可以使用数据绑定。

在 HyperLink 控件中有 3 个重要的属性。

- Text：一个可以使用数据绑定动态改变的值，可以使用{0}来表示 Text 属性的值。
- NavigateUrl：需要将链接转向的地址(URL)。
- Target：目标。选择链接的网页出现的位置。

在使用数据绑定的时候，可以通过改变 Text 属性所绑定的数据源的值来实现动态超链接。

4. 对控件进行定位

设计中，常常希望将控件拖放到一定的位置。可是当将控件拖入网页后，它的位置似乎就固定了。这是由于控件的排列顺序所决定的。

现在想改变控件的位置，只要先单击该控件，然后选择网页主菜单的【格式】|【位置】命令，弹出如图 8.3 所示的【定位】对话框。对话框中几个图标分别代表 CSS 定义的排列方式。其中：

① 代表默认的排列方式。即普通流(float)的排列方式。
② 代表靠左摆放(float: left)。
③ 代表靠右摆放(float: right)。
④ 代表绝对定位(absolute)。
⑤ 代表相对定位(relative)。



图 8.3　【定位】对话框

单击【绝对】(或【相对】)图标后，就可以改变控件的位置了。

5. 动态生成控件

Web(服务器)控件允许在程序的运行中动态生成，这是一项重要的优点。因为在某些情况下，动态生成控件比设计时创建的控件更加灵活。例如，在网页中进行搜索，要求用表格来显示搜索结果。因为事先不知道结果有多少项，因此表格的行数事先也不能确定，只能根据搜索结果动态生成。

动态生成控件的步骤如下。

(1) 放置新控件的容器以便给新创建的控件占位。如果没有容器时，也可以利用 Placeholder 或 Panel 控件来替代新控件的容器。

(2) 创建新控件对象。

(3) 给新控件设置属性。

(4) 将新控件加入到容器中来。

下面通过两个示例来说明创建的方法。

例 8.1 创建一个新的 Label 对象，Panel1 作为它的容器。

先拖入一个 Panel1 对象作为新控件的容器。然后给某个按钮编写以下代码。

```
Label myLabel = new Label();          //生成控件 myLabel1
myLabel.Text = "简单的动态 Label";    //给控件设置属性
Panel1.Controls.Add(myLabel);         //将新控件加入到容器中
```

例 8.2 动态创建一个表格。

在工具箱的 HTML 选项卡中提供了 Table 控件，在【标准】选项卡中同样提供了一个 Table 控件。两种控件各有特色。如果用于显示静态数据时采用 HTML 中的 Table 控件比较有利；如果表格需要动态生成时，使用【标准】选项卡中的 Table 控件比较有利。

首先应该明确的是，表(Table)是对象，表中的行(TableRow)也是对象，行中的列(单元格，TableCell)也是对象。父对象包括子对象，子对象又包括自己的子对象，每种对象都需要单独生成，然后组合到一起。

其中 Table 控件作为 TableRow 控件的父对象，支持名为 Rows 的属性，它是 TableRow 对象的集合。可以通过管理该集合(在其中添加或删除项)来指定表的行。TableRow 对象又支持名为 TableCell 对象的集合。

表中显示的内容将添加到 TableCell 对象中。单元格有 Text 属性，可以将其设置为任何 HTML 文本。具体步骤如下。

(1) 在窗体页中放置新控件的容器。在这里，Table 控件就是新控件的容器。将 Table 控件拖入窗体页中，设置好整个表的外观属性，比如 Font、BackColor 和 ForeColor 等。默认情况下 TableRow 控件和 TableCell 控件也将支持这些属性，当然也可以重新为个别行或单元格指定另外的外观属性，新设置的属性将覆盖父表中的设置。

(2) 可以将数据绑定到控件上，通常是向表添加 TableCell 控件。然后将单个 TableCell 控件的 Text 属性绑定到数据上，或者向单元格添加数据绑定控件(比如 Label 控件或 TextBox 控件)。

添加行的方法是

```
TableRow tRow = new TableRow();        //生成行对象
Table1.Rows.Add(tRow);                 //将行对象加入到表中
```

添加单元格的方法是

```
TableCell tCell = new TableCell();     //生成单元格对象
tRow.Cells.Add(tCell);                 //将单元格加入到行中
```

向单元格添加内容有多种方法，如表 8.2 所示。

表 8.2 向单元格添加内容的方法

添加的内容	操作方法
静态文本	设置单元格的 Text 属性

续表

添加的内容	操作方法
控件	生成一个控件实例，然后将其添加到单元格的 Controls 集合中
既有文本又有控件	通过创建 LiteralControl 类的实例来声明文本。像处理其他控件一样将该实例添加到单元格的 Controls 集合中

现在综合前面的方法动态生成一个表格。表格的行与列的数目是由两个文本框(TextBox1、TextBox2)中的数字决定。每个单元格中以静态文本形式显示其行号和单元格号。

先将标准 Table 控件拖入窗体作为动态表格的容器，并为该控件设置相关属性，然后在按钮的 Click 事件中编写如下代码。

```
public void Button1_Click (object sender, System.EventArgs e){
   int rowCnt;       //行的数目
   int rowCtr;       //当前行
   int cellCtr;      //每行包括的列数
   int cellCnt;      //当前的列
   rowCnt = int.Parse(TextBox1.Text);
   cellCnt = int.Parse(TextBox2.Text);
   for(rowCtr=1; rowCtr <= rowCnt; rowCtr++) {
   // 创建新行并加入到表中
     TableRow tRow = new TableRow();
     Table1.Rows.Add(tRow);
     for (cellCtr = 1; cellCtr <= cellCnt; cellCtr++)
      {
     // 创建新列并加入到行中
        TableCell tCell = new TableCell();
        tCell.Text = "行 " + rowCtr + ", 列 " + cellCtr;
        tRow.Cells.Add(tCell);
      }
    }
  }
```

显示结果如图 8.4 所示。



图 8.4　动态生成的数据表

有时，在数据表的某些单元格中放置控件(例如 TextBox)更加有利于数据绑定。此时

应先生成一个控件对象，然后将其添加到单元格的 Controls 集合中。例如：

```
TextBox Text1 = new TextBox();        //生成 Text1 对象
TableCell tcell = new TableCell();    //生成单元格对象
tcell.Controls.Add(Text1);            //将控件对象加入到单元格中
```

本示例中涉及一些数据类型的转换。在程序中这种转换是会经常遇见的。

8.2 ASP.NET 的事件处理模型

ASP.NET 是一个分布式系统，在这个系统中发生了事件，应该在什么地方处理？一些涉及数据改变的事件(例如访问或编辑数据库)当然只能由服务器处理。除此之外，在浏览器端发生的事件，有些可以由浏览器处理，也可以由服务器处理。这种情况下，究竟由谁处理更加有利呢？从理论上来讲，除了极少数几种事件以外，服务器几乎可以处理任何类型的事件，但是如果将一些事件放在浏览器中处理，反应会更灵敏，效率也会更高一些。因为用服务器处理事件时，需要首先将事件的信息传送到服务器，待服务器处理完毕以后再返回到浏览器，比浏览器处理多一个信息的往返过程。但是由浏览器处理事件的最大问题是处理的能力受限于浏览器本身。

ASP.NET 对事件处理的原则是：

- 基于服务器处理事件。
- 尽量减少事件处理中信息往返的次数。
- 调用浏览器执行辅助功能。

8.2.1 基于服务器的处理模型

ASP.NET 采用的是基于服务器处理的模型。这就是说，系统运行中发生的事件，不论发生在服务器端还是发生在浏览器端，基本上都由服务器进行处理。

ASP.NET 功能强大的根本原因在于拥有一个强大的.NET 框架服务器平台，此平台不仅提供了强大的类库，还提供了各种完善的服务。程序将在平台的监控下运行，运行完后还能根据浏览器的类型智能地返回不同的信息。因此由服务器处理事件不仅能力强，而且更加安全。

另一方面，从浏览器的发展趋势来看，在不远的将来，必然会有更多的智能终端为了不同的目的，利用有线、无线等方式通过宽带网连入到因特网中。这些设备中相当一部分将变得更加小巧和更加多样化。这些设备处理事件的能力各不相同，有的可能会变得越来越弱，甚至可能根本不具备处理事件的能力。这种情况下，强调用服务器(而不是用浏览器)来处理事件是合适的。

ASP.NET 的目标是创建下一代网络应用的平台，基于上述的发展趋势，采用基于服务器处理的模型符合确定的目标。

8.2.2 尽量减少信息的往返次数

为了减少事件处理中信息往返的次数，系统采用了以下策略，即客户端发生的事件，并不是每发生一次就向服务器传送一次信息。默认情况下，只有当服务器端按钮(Button)

被单击时，才集中向服务器传递事件信息。其他支持改变(Change)事件的服务器端控件，如文本框、下拉列表框、单选按钮、复选框等，当它们的 Change 事件发生时，先将事件的信息暂时保存在客户端的缓冲区中，等到下一次向服务器传递信息时(单击按钮时)，再和其他信息一起发送给服务器。以减少传送信息的频度。

如果有的控件的 Change 事件需要立即得到响应时，只需要将该控件的 AutoPostBack 的属性设为 true 即可。值得注意的是，这种设置不宜过多，如果过多有可能降低系统的运行效率。

当服务器同时收到多个事件信息时，对 Change 事件的处理总是放在其他事件之前，而对其他事件的处理顺序则是不确定的。

8.2.3　结合浏览器处理事件

1. 有的事件只能由浏览器处理

有的事件只能由浏览器处理。例如鼠标移动(MouseMove)或某些动态图形所引发的事件等，由于这些事件发生得过于频繁，没有必要也不允许传送到服务器去处理，这些事件只能在浏览器端进行处理。

2. 调用浏览器执行辅助功能

有些事件虽然服务器能够完成处理工作，但是如果再调用浏览器的 DHTML 执行某些辅助功能，执行的效率会更高，功能将变得更加丰富。为了实现这一功能，需要在服务器控件的定义中增加少量代码。下面举例说明。

例 8.3　删除记录前的进一步确认。

在浏览器端通过调用 JavaScript 脚本的 alert() 或 confirm()方法可以打开两种提示对话框。

- alert()方法输出的警告对话框：对话框中主要包括一个提示标记、一个提示语句和一个按钮，其主要作用在于提示客户进行某些操作。语句 alert ("请输入姓名！") 输出的窗口如图 8.5(a)所示。
- confirm()方法输出的确认对话框：对话框中主要包括一个标记、一条提示语句和两个按钮。其主要作用是确认某种行为。语句 confirm ("真的准备删除吗？")输出的对话框如图 8.5(b)所示。




(a) 警告对话框



(b) 确认对话框

图 8.5　两个 DHTML 的提示对话框

现在在窗体页中放入一个删除记录的按钮(Button1)，这是一个服务器控件。运行这个控件以前，最好先确认一下，这个确认过程最好放在浏览器端，调用 DHTML 功能来

完成。

下面就是实现上述功能的相关代码。在这里重点看一下服务器控件是如何调用 DHTML 功能的。

```
<%@ Page Language="C#" %>

<script runat="server">
  void Page_Load(object sender, EventArgs e)
    {
     Button1.Attributes.Add("onclick", "javascript:return confirm('真的准
     备删除吗？')");
    }
  protected void Button1_Click(object sender, EventArgs e)
   {
     Response.Write("执行删除任务.");
   }
</script>

<body>
    <form id="form1" runat="server">
    <div>
        <asp:Button ID="Button1" runat="server" Text="Button"
         OnClick="Button1_Click" /></div>
    </form>
</body>
</html>
```

程序运行时，如果单击 Button1 按钮，将先弹出确认对话框，若单击对话框中的【确定】按钮，将执行 Response.Write("执行删除任务.")语句；若单击【取消】按钮，将不执行上述输出任务。

调用 DHTML 功能是在网页的 Page_Load 事件中通过以下语句设置的。

```
Button1.Attributes.Add("onclick", "javascript:return confirm('真的准备删除
吗？')");
```

其中“Attributes.Add()”给按钮附加了调用浏览器的 DHTML 功能。

例 8.4 自动显示被选中的服务器图形按钮。

若界面上有多个服务器图形按钮，分别用于执行不同的任务。为了防止单击错误，最好当鼠标指针移动到某个按钮上时，先用 DHTML 改变该按钮的外形，来作为该按钮已被选中的信号。

为了简单起见，现在用一个图形按钮(ImageButton1)来说明设计方法。这个图形按钮是一个服务器控件，如果单击该按钮，将调用服务器方法 Response.Write()来显示“执行 1 号任务！”字符串。现在要求当鼠标指针移动到按钮上方(不单击按钮)时，更换按钮上的图像(假定原来的 photo1.jpg 换成 photo2.jpg)。当鼠标指针离开时还原为原来的图像。

网页中的相关代码如下。

```
<%@ Page Language="C#" %>
  <script runat="server">
```

```
  protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
   {
    Response.Write("执行 1 号任务！");
   }
  </script>

<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
        <asp:ImageButton ID="ImageButton1" runat="server" ImageUrl="photo1.jpg"
         OnClick="ImageButton1_Click" onmouseover="this.src='photo2.jpg'"
            onmouseout="this.src='photo1.jpg'" EnableViewState="False"/>
    </div>
  </form>
 </body>
</html>
```

其中，定义按钮的代码：

```
<asp:ImageButton ID="ImageButton1" … onmouseover="this.src='photo2.jpg'"
         onmouseout="this.src='photo1.jpg'"…/>
```

粗体表示的部分就是调用浏览器端 DHTML 功能。即当鼠标指针移动到控件上方时启动 onmouseover 事件，将图形按钮上的图像改成 photo2.jpg；当鼠标指针离开按钮时启动 onmouseout 事件，将图形还原成 photo1.jpg。

3. 调用浏览器的其他功能

ASP.NET 提供的某些服务器控件，如校验控件、TreeView 控件等，能够自动对客户传来的信息进行检测，判断其是否具有足够的 DHTML 功能(通常是指 IE 的版本是否在 5.0 以上)。如果具备这些功能，将会根据事先的设定自动将部分辅助功能分配在浏览器端执行，以提高程序的运行效率。具体情况将在后续章节中讲述。

8.3　应 用 示 例

下面是 6 个应用示例，这些示例虽然简单，但每个示例代表一种不同的类型。通过这些不同类型的示例可以进一步理解常用控件的使用方法及事件模型的执行过程。

例 8.5　两种事件处理代码的比较。

假定在网页中要写一段问候程序。这段程序可以在服务器端执行，也可以在浏览器端执行。为了便于比较，在创建 ASPX 网页时，采用单文件存储模式。现在编写服务器端执行的代码如下。

```
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
   {
```

```
        int tt1=int.Parse(DateTime.Now.Hour.ToString());
        if ((tt1 >= 6) && (tt1 <= 12))
          {
           Response.Write("上午好!");
          }
        else if ((tt1 > 12) && (tt1 <= 18))
         {
          Response.Write("下午好!");
         }
        else
          Response.Write("晚上好!");
     }
    </script>
```

这段代码中<script runat="server">…</script>，表明它是在服务器端执行的代码。代码首先取出当天的时间(小时)，将其转换成整型数，然后进行判断。如果是上午(6～12 点)，显示“上午好!”；如果是下午(12～18 点)，显示“下午好!”；其他时间显示“晚上好!”。这段代码本身没有问题，但不符合实际需要。因为全世界的时区各不相同，各个客户所在的时区都不同。应该根据客户所在位置的时间确定问候的内容，因此这段代码应该在浏览器端执行。

在浏览器端执行的代码如下。

```
<script language=javascript >
  {
   var now=new Date();
   var tt1 = now.getHours();;

   if ((tt1 >= 6) && (tt1 <= 12))
     {
       document.write("上午好!");
     }
   else if ((tt1 > 12) && (tt1 <= 18))
     {
      document.write("下午好!");
     }
   else
      document.write("晚上好!");
  }
</script>
```

注意和上一段代码相比，有以下区别。

- 代码用<script language=javascript >…</script>括起来，这表明这段代码在浏览器端执行(去掉了 runat="server"属性)，而且使用的语言是 JavaScript。
- 代码中不能使用服务器端对象 Response，而必须改用浏览器端函数 document.write ()。

例 8.6 一个简单的学生选课系统。

本示例的主要目的是学会开发一个应用程序。要求先填完姓名、学号，然后在下拉列表中选课。当单击按钮时，将提示姓名、学号以及选择的课程。

创建的步骤如下。

(1) 建立一新项目，从工具箱向窗体拖入以下 Web 控件：两个 TextBox、3 个 Label、一个 Button 和一个 DropDownlist。界面的部署如图 8.6 所示。

(2) 单击 DropDownList 控件属性 Items 右边的省略号按钮，弹出【ListItem 集合编辑器】对话框，如图 8.7 所示。

(3) 分别在 Text 及 Value 中添加下拉列表的选项。

(4) 在【设计】视图中双击按钮图标，打开代码(隐藏)文件，在按钮单击事件中编写事件处理如下代码。

```
private void Button1_Click(object sender, System.EventArgs e)
  {
    Response.Write(TextBox1.Text+" 学号："+TextBox7.Text+" ");
    Response.Write("你选择的课程是：  "+ DropDownList1.SelectedValue);
  }
```

代码中 DropDownList1.SelectedValue 代表下拉列表中选项的值。




图 8.6　学生选课系统

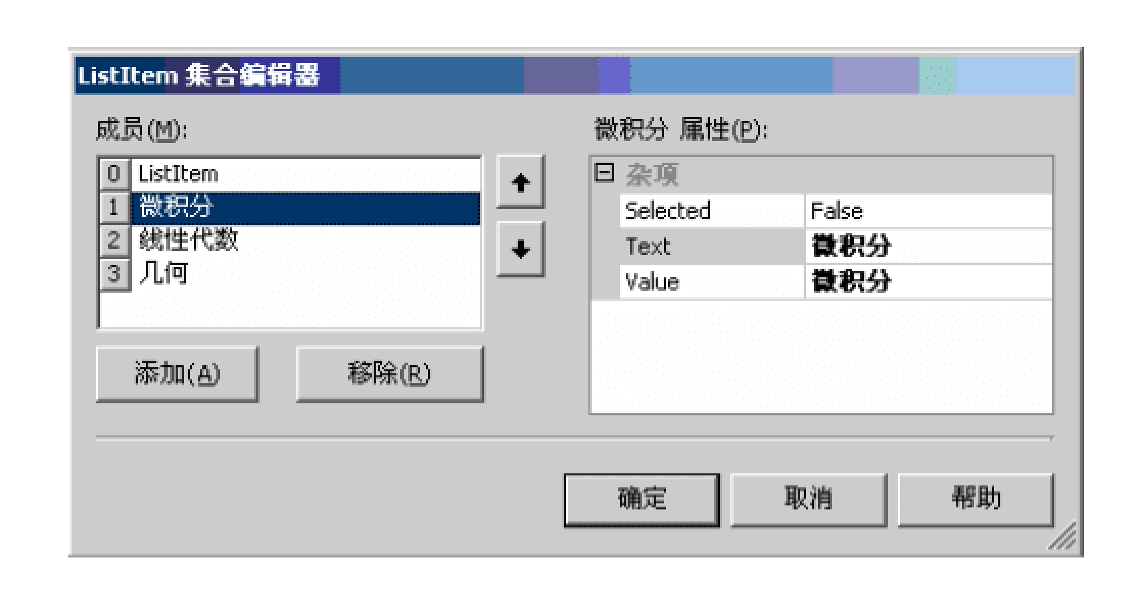


图 8.7　【ListItem 集合编辑器】对话框

例 8.7　摄氏、华氏温度的转换。

本示例的主要目的是掌握不同类型数据的转换方法。

温度有两种表示方法：摄氏和华氏。设 C 代表摄氏，F 代表华氏。两种温度的转换公式如下。

华氏转换摄氏时为

```
C=(F - 32) * 5/9
```

摄氏转换华氏时为

```
F = 9/5 * C + 32
```

现在设计一网页完成两种温度的转换工作。

利用标准控件设计程序的界面。该界面使用了两个 Label、两个 TextBox、一个 RadioButtonList、一个 Button。具体设置如图 8.8 所示。

给 RadioButtonList 的 Items 属性设初始值。方法是单击 Items 属性的省略号按钮，将弹出如图 8.9 所示的对话框。

在【ListItem 集合编辑器】对话框中添加两个成员(“转换成摄氏”和“转换成华氏”)，分别为成员的 Text 和 Selected 属性设初值。

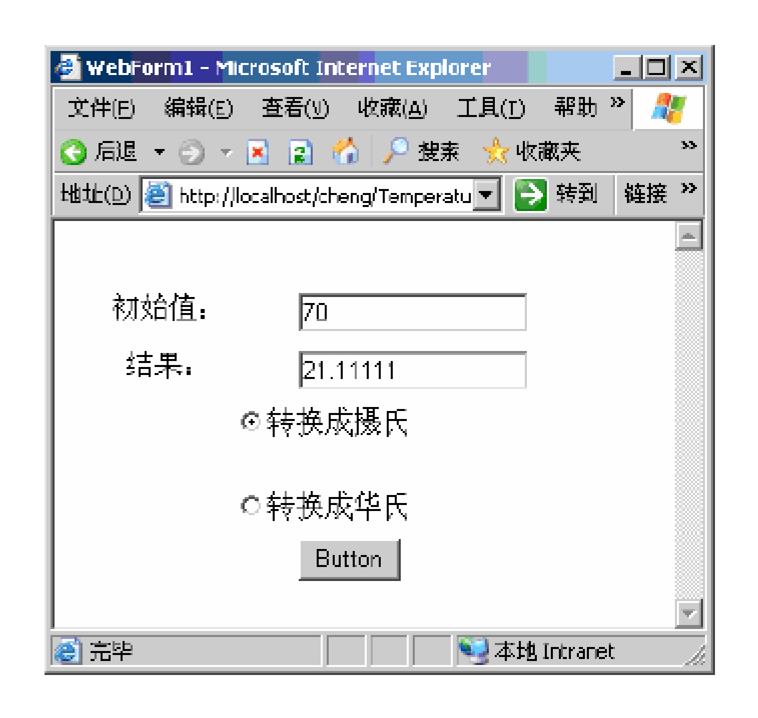


图 8.8 温度转换界面

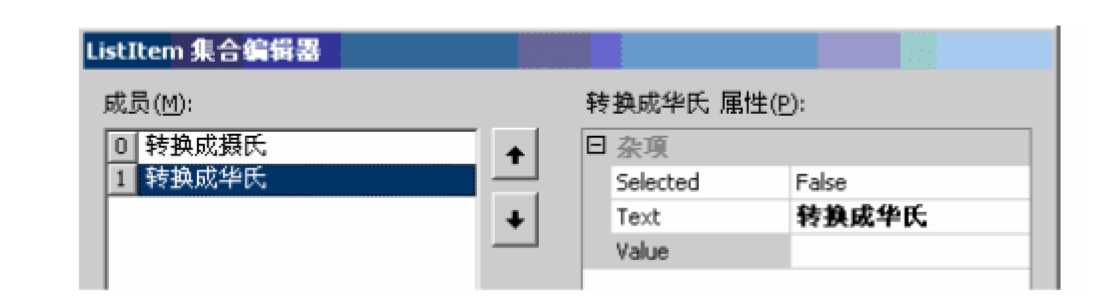


图 8.9 RadioButtonList 的 Items 属性设置

在按钮的 Click 事件中编写代码。该代码完成以下任务。

(1) 如果 TextBox1 为空时，提示给温度赋初值。

(2) 换算前先判断 RadioButtonList 中的选项，并按照公式进行换算。

(3) 将换算结果显示在 TextBox2 中。

具体代码如下。

```
private void Button1_Click(object sender, System.EventArgs e)
  {
    float rr=0;
    if(TextBox1.Text=="")
      Response.Write("请输入初始值. ");    //提示输入
    else
   {
     //换算
    if (RadioButtonList1.SelectedIndex==0)
      rr=((float.Parse(TextBox1.Text)-32)*5/9);
    else
      rr=(float.Parse(TextBox1.Text)*9/5+32);
    TextBox2.Text=rr.ToString();    //显示结果
    }
 }
```

类库中的 Math 类是一个用于数学运算的类，包括有大量的数学运算的静态方法，可以通过类名直接调用。比如现在只想使结果保持一定的小数位时，可以调用该类的 Round()方法。例如下述语句：

```
TextBox2.Text = Math.Round(rr,2).ToString();
```

就是将结果(rr)只保留两位小数，对第三位进行四舍五入。

以上程序是在按钮的 Click 事件中进行换算的，这是通用的办法。如想改变换算的时机，在改变温度的初始值后立即进行换算，可以将 TextBox1 控件的 AutoPostBack 属性改为 true，并将换算的代码写在该控件的 TextChanged 事件中。

例 8.8 多图片之间切换。本示例的主要目的是学会图形控件的使用方法。

要求设计一个由 RadioButtonList 控件选择显示不同图片的界面，如图 8.10 所示。

设计步骤如下。

(1)　右击网站名，在弹出的快捷菜单中选择【添加】|【添加现有项】命令，将多张图片(设图片的名称为 cus1.bmp、cus2.bmp、…、cus7.bmp 等)加入到项目中。

(2)　将 RadioButtonList 及图形控件(Image)和按钮控件(Button)从工具箱拖入窗体，单击 RadioButtonList 控件的属性 Items 右边的省略号按钮，弹出【ListItem 集合编辑器】对话框，如图 8.11 所示。

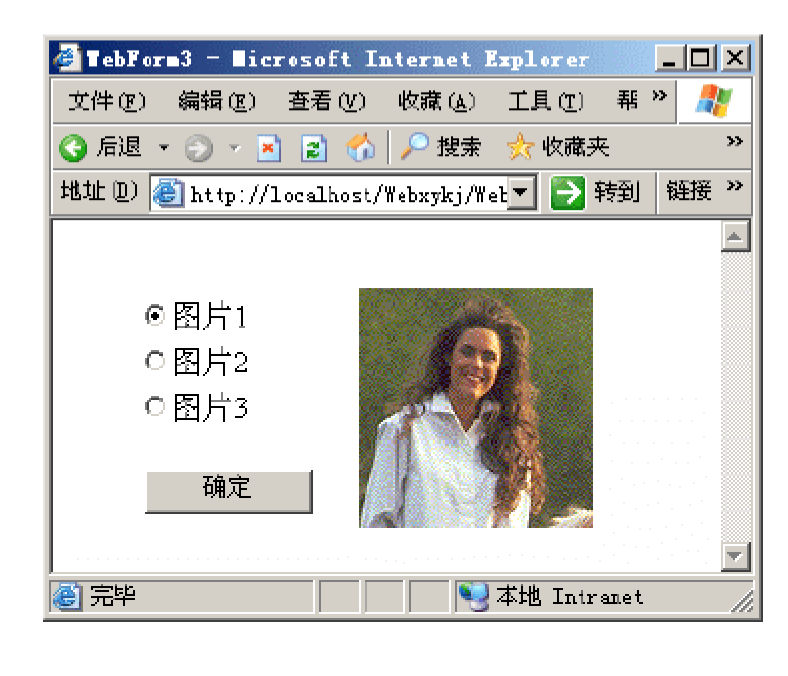


图 8.10　图片选择

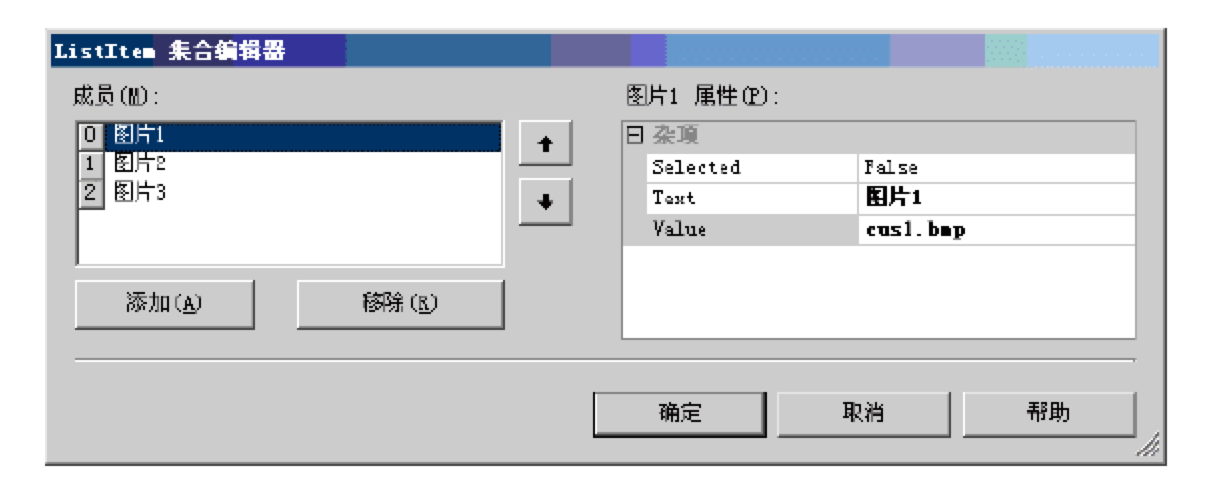


图 8.11　【ListItem 集合编辑器】对话框

(3)　在对话框中添加项，为每项的 Text、Value 属性赋值，如图 8.11 所示。

(4)　打开 RadioButtonList 控件的 SelectIndexChanged 事件，编写如下代码。

```
private void RadioButtonList1_SelectedIndexChanged(object sender,
System.EventArgs e)
  {
   Image1.ImageUrl=RadioButtonList1.SelectedValue;
  }
```

(5)　运行程序后，改变 RadioButtonList 控件的选择，然后单击按钮，图形控件中的图片将跟随改变。

如果想在改变 RadioButtonList 控件的选择时立即得到响应，不需要单击按钮，将 RadioButtonList 控件的 AutoPostBack 属性设置为 true 即可。

例 8.9　页面跳转。本示例的主要目的是学会利用 Response 对象进行页面跳转的方法。

在网页的应用中很多地方需要进行页面跳转。前面已经学过利用超链接方法切换页面，这里将要学习另一种页面切换的方法。这是一种在服务器端利用 Response 对象进行页面切换的方法。在 ASP.NET 中使用这种方法切换页面的机会是很多的。

Response 对象是 HttpResponse 类的实例，只不过客户可以直接使用罢了。Response 对象包含很多属性和方法。例如 Response.Write(字符串);可以向浏览器送去字符串。本示例中将要用到 Redirect(字符串)方法，其中字符串参数就是新 URL。这个方法的作用是将客户端重新定位到新的 URL。

实现步骤如下。

(1)　设置两个以上的窗体(如 WebForm1.aspx、WebForm2.aspx、…、WebForm7.aspx、…)。在其中之一放置标准控件，设有 Label、DropDownList、Button 各一个，如图 8.12 所示。

(2)　单击 DropDownList1 控件，再单击属性 Items 右边的首略号按钮。在弹出的对话

框中添加需要转换的网页。如果是本项目中的网页，只需在 Value 栏中填写网页名即可(例如 WebForm7.aspx)。如果是项目以外的网页，需要写出完整的 URL 地址，如图 8.13 所示。

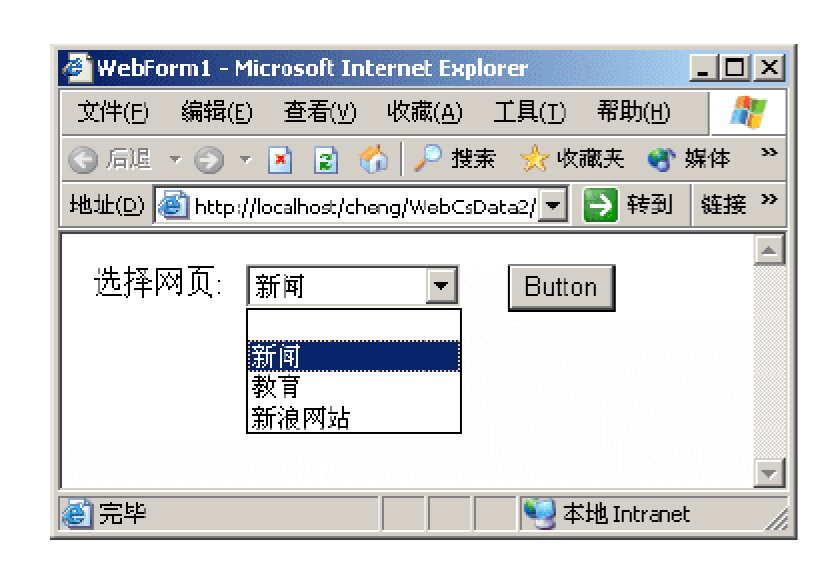


图 8.12　页面跳转界面

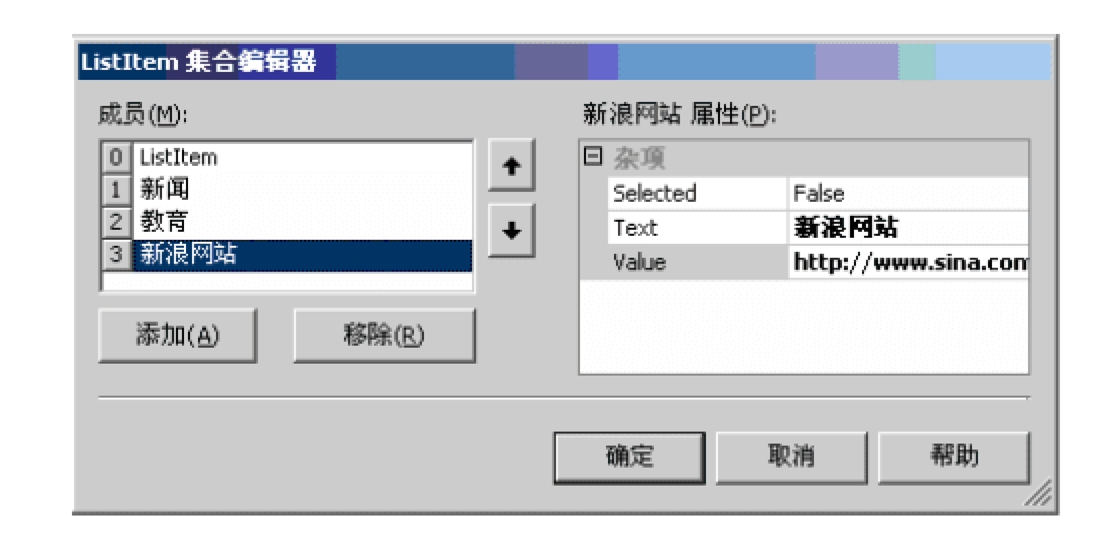


图 8.13　调用网站外的网页时的设置

(3)　在 Button 的 Click 事件中编写如下代码。

```
private void Button1_Click(object sender, System.EventArgs e)
 {
   Response.Redirect(DropDownList1.SelectedItem.Value);
 }
```

如果想在改变选择时立即得到响应，除了需要将 DropDownList1 控件的 AutoPostBack 属性改为 true，还要改变选择的事件，将 Button1_Click 改变为 DropDownList1。

例 8.10　动态公告条。在本示例中使用了 Adrotator 控件，并结合使用了 XML 文件。

AdRotator 控件用于显示公告栏中的某一公告，此公告栏由一个基于 XML 的公告文件指定。该文件中包括<Ad>节点，节点中包括公告所在图像的路径(URL)、图像不存在时显示的文本以及公告显示次数占总次数的比率(百分比)。由于各个公告的重要性不同，应给各个公告设置不同的显示概率，使得重要的公告显示的次数多于一般的公告。

下面结合示例说明使用的方法，如图 8.14 所示。

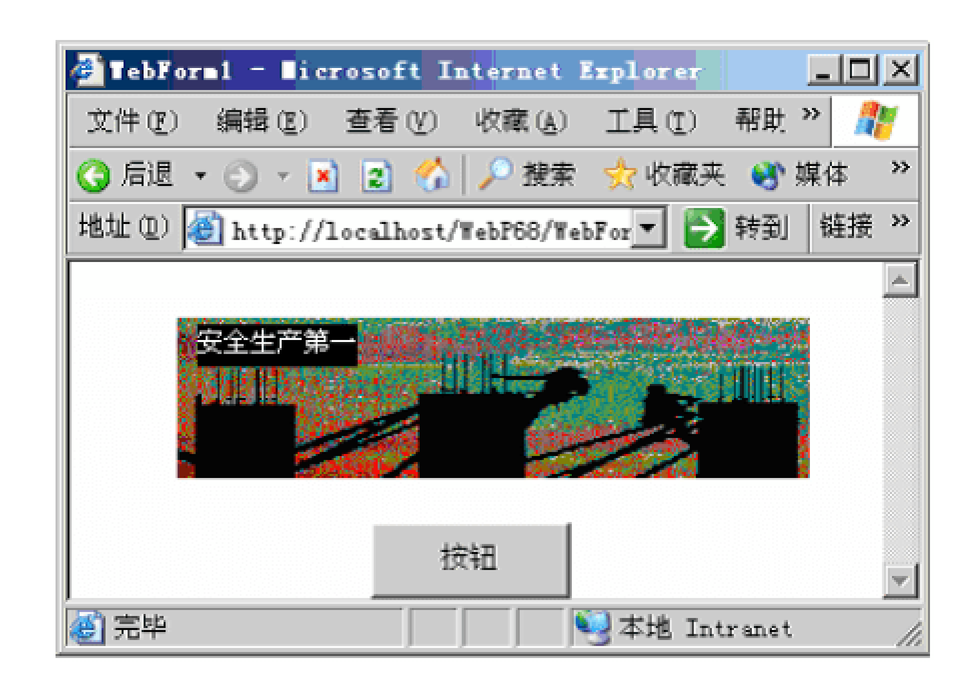


图 8.14　动态公告条界面

具体操作步骤如下。

(1)　创建一项目，将多个公告图片放入项目中。

(2)　右击项目名，在弹出的快捷菜单中选择【添加】|【添加新项】命令，在【模板】对话框中打开 XML 文件。

(3) 按照下面的格式编写 XML 文件。

```
<Advertisements    xmlns="http://schemas.microsoft.com/AspNet/AdRotator-
Schedule-File">
  <Ad>
    <ImageUrl>CHIP.jpg</ImageUrl>
    <Keyword>连接到芯片图</Keyword>
    <Impressions>20</Impressions>
    <NavigateUrl>WebForm7.aspx</NavigateUrl>
  </Ad>
  <Ad>
    <ImageUrl>CONSTRUC.jpg</ImageUrl>
    <Keyword>连接到基建图</Keyword>
    <Impressions>30</Impressions>
    <NavigateUrl>WebForm7.aspx</NavigateUrl>
  </Ad>
  <Ad>
    <ImageUrl>SKYLINE.jpg</ImageUrl>
    <Keyword>连接到工厂外景图</Keyword>
    <Impressions>15</Impressions>
    <NavigateUrl>WebForm7.aspx</NavigateUrl>
  </Ad>
</Advertisements>
```

其中：

- <ImageUrl>…</ImageUrl>标记的中间为图片名。
- <Keyword>…</Keyword>标记的中间为关键字，用来说明公告的特点。如果不能显示图片时，显示这些关键字。
- <Impressions>…</Impressions>中间为一个大于 0 的整数，代表显示的概率，数字越大显示的次数越多。
- <NavigateUrl>…</NavigateUrl>代表网页的 URL。

(4) 将文件以后缀为.xml 的名字存储，并在 AdRotator 控件的 AdvertismentFile 属性中填入上述文件名。

(5) 运行程序，以检查效果。反复单击按钮，每单击一次就相当于访问网页一次。查看各图片出现的次数是否符合前面设置的比例。

8.4 小　　结

控件实质上就是一个类，一种可视化的类。利用控件进行设计，可以起到“所见即所得”的直观效果。

事件处理模型是影响系统运行方式的重要方面。事件处理模型有多种方式，有的以浏览器处理为主，有的以服务器处理为主。就好比人们到超市去购货，可以使用现金也可以使用信用卡，前一种方式是基于浏览器的处理方案，因为此时所有计算都在现场完成；而后一种方式是基于服务器的处理方案，因为此时客户只需要刷卡，所有财务方面的问题都由银行系统的服务器自动完成。也可以用刷卡方式，但用一点零钱(现金)作为调整，这就

是服务器结合浏览器的方式。

在传统的 HTML 网页或 ASP 网页中，当在浏览器端下载网页以后，发生的事件都在浏览器端处理，直到提交网页时才再度与服务器进行交互。现在 ASP.NET 3.5 采用的是基于服务器的事件驱动模型，程序运行中浏览器与服务器之间的交互变得更加频繁，这不仅充分发挥了.NET 框架平台的作用，也使得系统的运行方式更加接近于桌面系统，使得两种设计思想更加趋于一致，从而使得广大程序设计者学习和掌握 ASP.NET 3.5 系统变得更加容易。

8.5 习　　题

1. 填空题

(1) 当需要将 TextBox 控件作为密码输入框时(要求隐藏密码的代码)，应该将控件的属性__________设置为 Password。

(2) 当一个 Web 控件上发生的事件需要立即得到响应时，应该将它的________属性设为 true。

(3) 下面是一个转移到新网页的指令。

```
Response.________ ("新网页的Url");
```

(4) 将下列数据(nn)在 TextBox 控件中显示出来。

```
double nn = 4512.65;
TextBox1.Text  =______________________;
```

(5) 将下列字符串转换为浮点类型的数据，以便进行计算。

```
string ss = "4109.54";
double dd =______________________;
```

2. 选择题

(1) 下面几个图形控件中，不能执行鼠标单击事件的控件是________。

A. ImageButton　　B. Image　　C. ImageMap

(2) 当需要用控件来输入性别(男、女)或婚姻状况(已婚、未婚)时，为了简化输入，应该选用的控件是______。

A. RadioButton　　B. CheckBoxList

C. CheckBox　　D. RadioButtonList

3. 判断题

(1) HTML 控件属于浏览器控件，不接受服务器的控制。　　(　　)

(2) HTML 控件与 HTML 元素一一对应，而 Web 控件的抽象程度更高，一个控件设置不同的属性时可以实现不同的功能。　　(　　)

(3) Web 控件中的几个按钮都可以起到向服务器提交数据的作用。　　(　　)

4. 简答题

(1) 在 ImageMap 控件中如何实现分区超链接?

(2) 如何对控件进行定位?

(3) 简述 ASP.NET 的事件模型。

(4) 举例说明 Web 控件调用 DHTML 辅助功能的方法。

(5) 举例说明动态创建数据表的方法。

(6) 简述系统在执行 HTML 表单与 Web 表单时的区别。

(7) HTML 表单中的两个属性 Active 与 Method 各起什么作用?

5. 操作题

(1) 创建多张网页，并实现网页之间多种方法的转移。转移时使用的方法包括:

- 利用 HTML 表单。
- 利用 HTML<a…>超链接标记。
- 利用 HyperLink 控件。
- 利用 Response.Redirect()方法。

(2) 输入两个数值，其中一个是分子，另一个是分母。输出两数之比的百分比。

(3) 在 ImageMap 控件上放置一张地图，实现对各个地区的超链接。

第 9 章 状态管理

网站与桌面系统的工作方式不同，与传统的分布式系统也不相同。在桌面系统中，系统资源被独占；在传统的分布式系统中，资源虽然分布在系统的各个环节，但是只要不专门指定，系统总是保持连接的。网站系统虽然也是一个分布式系统，但由于服务器要为众多的客户服务，浏览器与服务器之间的连接是不连续的，状态也是不保持的，HTTP 是一个无状态的通信协议。这就是说，在网站系统中，每次浏览器与服务器之间的连接都是暂时的。当浏览器与服务器之间的一次会话结束，它们之间的连接也就自动断开了，下一次会话与本次连接无关，两次连接之间不存在任何联系。

为什么不保持状态呢？这是因为访问网站的客户常常川流不息。如果要求系统将所有被访问的网页的状态都记忆下来，必然会耗费大量的系统资源，严重地降低程序的运行效率。

然而，在网站应用中有的状态却是需要保留的，比如客户在购货车中订购商品、客户登录的身份、对问卷调查所作的回答等，这些状态中有的希望能够保留一定的时间，以便联合处理或者在一定的范围内进行传递和共享。

为此，系统提供了状态管理方法，允许有选择地将一些状态(数据)在一定的时间内持续地保存下来。本章将要介绍几种状态的管理方法，具体内容包括:

- 状态的类型。
- 视图状态。
- 应用程序状态。
- 会话状态。
- Cookie 状态。
- 简单的应用示例。
- Web 窗体页的生命周期。

9.1 状态的类型

ASP.NET 提供了 4 种状态类型，分别应用于不同的目的。

- 视图状态：用于保存本窗体页的状态。
- 应用程序状态：用于保存整个应用程序的状态，状态存储在服务器端。
- 会话状态：用于保存单一客户的状态，状态存储在服务器端。
- Cookie 状态：用于保存单一客户的状态，状态存储在浏览器端。

9.2 视图状态

简单地说，视图状态就是本窗体的状态。保持视图状态就是在反复访问本窗体页的情

况下，能够保持状态的连续性。

微软创建 ASP.NET 时追求的目标之一，是尽量使网站的设计与桌面系统一致。ASP.NET 中的事件处理模型是实现本目标的重要措施(参见第 7 章)，该模型是基于服务器处理事件的，当服务器处理完事件后通常再次返回到本窗体以便继续后续的操作。如果不保持视图状态，就是说当窗体页返回时，窗体页中原有的状态(数据)都不再存在，这种情况下怎样能够继续窗体的操作？

下面用一个简单的示例来说明这种情况。

假定向窗体中放入几个 HTML 控件(浏览器端控件)。

- 一个 Input(Text)控件：用来输入姓名。
- 一个 Input(Password)控件：用来输入口令。
- 一个 Input(Text)控件：用来输入数量。
- 一个 Input(Submit)按钮控件：用来向服务器提交数据。

界面如图 9.1 所示。



图 9.1　使用 HTML 控件的示例

当输入完数据，单击【提交】按钮时，提交数据的同时，网页被重新启动，网页中原有数据都不见了。这就是不保持视图的结果。若直接改用标准控件，再按照前面的方法操作，当单击【提交】按钮提交数据后状态仍然保持。

注意：用于口令的输入框是一个例外，它不能保持状态。

系统是利用什么方法来保持视图状态的呢？原来在这里微软采用了一种比较特殊的方式，只要在浏览器中打开网页的源文件来查看一下，就会发现在源代码中已经自动增加了一段代码。例如：

```
<input type="hidden" name="__VIEWSTATE"
   value="dDwtNTMwNzcxMzI0Ozs+4G+LcyJuZIyhKS53MRLvCJ46CUk=" />
```

这说明在网页中已经自动增加了一个隐含控件，控件的名字为“__VIEWSTATE”。由于这个新控件是隐含控件(type="hidden")，因此增加它并不会改变界面上的布局。控件中的 value 属性就是窗体页中各个控件以及控件中的数据(状态)。为了安全，这些数据在默认情况下都使用散列码(Hash Code)进行了加密，已经变得难以辨认。当网页提交时，浏

览器端首先将当前网页中的各种状态保留到这个字段中，当网页返回到本窗体页时，再自动把这些状态反馈给返回的窗体页，也就恢复了窗体页中各控件的状态。

视图状态只能在本网页与服务器的往返中保持，而不能在不同网页之间传递，这是和其他状态所不同的地方。在默认情况下几乎所有服务器控件都具有保持视图状态的功能。

保持视图状态带来了好处，但同时也带来一些新的矛盾。如果控件中包括的数据量很大(例如某控件内有数百条记录)时，将会延长网页往返时需要的时间。ASP.NET 对网页在往返时数据的转换进行了优化，从而大大减少了转换的时间。

由于使用了散列码(一种加密的校验码)，视图状态的安全已经可以满足大多数客户的要求。如果对某张网页还有更高的安全要求时，可以对视图状态进一步加密。即在网页的<%@ Page…%>增加以下设置。

```
<%@ Page ViewStateEncryptionMode = "Always" %>
```

也可以在网站的配置文件中设置 ViewStateEncryptionMode 属性，以便对网站的多张网页实现视图状态的加密。方法如下。

```
<configuration>
  <system.web>
    <pages ViewStateEncryptionMode = "Always" />
  …

   </system.web>
</configuration>
```

提示：如果不是特别需要，不要对视图状态进一步加密，因为这种加密处理是以性能的损失为代价的。

9.3 应用程序状态

Application 对象是 HTTPApplicationState 类的实例。Application 是属于全局性的对象，用于存放应用程序中多个客户共享的信息。当客户第一次访问某虚拟目录的资源时被创建，退出应用程序或关闭服务器时被撤销。

Application 对象利用“键—值”对的字典方法来定义，其中“键”为字符串，代表状态的“名”，“值”可以是任何类型的数据。例如：

```
Application["Message"] = " MyMsg";
                 // 给名为 Message 的 Application 对象赋值，值为 MyMsg
string Myvar= (string)Application["Message"];
                 //  取出名为 Message 的 Application 的值赋给字符串 Myvar
```

为了和以前的 ASP 版本兼容也可以使用以下语句。

```
Application.Contents["Message"] = " MyMsg";
string Myvar=(string) Application.Contents["Message"];
```

可以利用 Application 的 Add 方法向 Application 的集合中添加项，也可以利用 Remove 方法删除不需要的项。例如：

```
Application.Add("Message","MyValue");
Application.Remove("Message");
```

可以利用 Clear()或者 RemoveAll()方法清除 Application 集合中的内容。例如：

```
Application.Clear();
Application.RemoveAll();
```

由于信息共享，有可能出现多个客户同时访问 Application 时而引发竞争。为了防止竞争带来的影响，可以利用 Application 对象的两个方法 Lock()和 UnLock()。其中 Lock()方法用于锁定对象，不允许其他进程访问；UnLock ()方法用于解锁，以便允许其他进程访问。

例如，将 Application["counter"]用来统计访问网站的人数时可以采用以下代码。

```
Application.Lock();             //锁定 Application 对象，避免多客户竞争访问
  Application ["counter"] = (int)Application ["counter"] + 1;
Application.UnLock();           //解除对 Application 对象的锁定
```

应用程序状态只能在网站运行时存在。如果 Web 服务器关闭或崩溃了，应用程序状态所保留的信息也会损坏或丢失。因此，对于那些需要永久保留的状态应当保存在数据库或其他永久性的存储器中。

9.4　会 话 状 态

9.4.1　概述

会话状态(Session State)是为单个客户保留的状态。在网站中，每一个新访问的客户都将产生自己的会话(Session)对象。这个 Session 对象在服务器端进行管理，只能为当前访问的客户服务。如果另一位客户也打开网站，他也将拥有自己的 Session 对象，两个客户的 Session 对象之间即使同名，也不能共享同一个 Session 对象。

在早期的 ASP 中，Session 只能用于存储和检索字符串信息。而在 ASP.NET 中增强了会话状态的功能。现在的 Session 已经成为一个带有方法和属性的真正的.NET 对象。可以在 Session 对象中存储任何数据类型，包括客户定义的类和结构。

Session 对象是 HttpSessionState 类的实例。HttpSessionState 类的公共属性如表 9.1 所示。

表 9.1　Session 对象的主要属性

属 性	说　明
CodePage	获取或设置当前会话的代码页的标识符
Contents	获取对当前会话状态对象的引用
Count	获取会话状态集合中的项数
IsCookieless	获取一个值，该值指示会话 ID 是嵌入在 URL 中还是存储在 HTTP Cookie 中
IsNewSession	获取一个值，该值指示会话是否与当前请求一起创建的
IsReadOnly	获取一个值，该值指示会话是否只读

续表

属 性	说 明
IsSynchronized	获取一个值，该值指示对会话状态值的集合的访问是否同步(线程安全)
Item	已重载。获取或设置个别会话值。 在 C# 中，该属性为 HttpSessionState 类的索引器
Keys	获取存储在会话中的所有值的键的集合
LCID	获取或设置当前会话的区域设置标识符(LCID)
Mode	获取当前会话状态的模式
SessionID	用于获取会话的唯一标识 ID
StaticObjects	获取由 ASP.NET 应用程序文件 global.asax 中的 <object Runat="Server" Scope="Session"/> 标记声明的对象的集合
SyncRoot	获取一个对象，该对象可用于同步对会话状态值的集合的访问
Timeout	获取并设置会话状态所允许的时限(以分钟为单位)

HttpSessionState 类的主要方法如表 9.2 所示。

表 9.2 Session 对象的主要方法

方 法 名	说 明
Abandon	取消当前会话
Add	将新的项添加到会话状态中
Clear	清除会话状态中的所有值
CopyTo	将会话状态值的集合复制到一维数组中
Equals(从 Object 继承)	已重载。确定两个 Object 实例是否相等
GetEnumerator	获取当前会话中所有会话状态值的枚举数
GetHashCode(从 Object 继承)	用做特定类型的哈希函数，适合在哈希算法和数据结构(如哈希表)中使用
GetType(从 Object 继承)	获取当前实例的类型
Remove	删除会话状态集合中的项
RemoveAll	清除所有会话状态值
RemoveAt	删除会话状态集合中指定索引处的项
ToString(从 Object 继承)	返回表示当前 Object 的 String

9.4.2 Session 对象中方法的调用

Session 对象的方法可以用来保存会话状态和管理会话状态两个方面。

1. 保存会话状态

保存 Session 对象时可以使用以下语句。

```
Session["Message"] = "MyMsg";
```

取出 Session 对象时可以使用以下语句。

```
string MyVar= (string)Session["Message"];
```

为了与 ASP 的早期版本兼容，也可以使用 Contents 属性访问这些值。语句如下。

```
Session.Contents["Message"] = "MyMsg";
string MyVar= (string)Session.Contents["Message"];
```

2. 启动会话状态

应用程序状态在网站中总是可用的，而会话状态在使用前必须先启动。不过，因为配置文件(Machine.config)的默认设置是启动会话状态，因此不需要额外的步骤就能启动它。虽然如此，还是应该知道，是 Machine.config 和应用程序的 Web.config 配置文件的设置决定了会话状态是被启动还是被禁止。

如果想延迟到需要时再启动会话状态，则可以在页面中编写以下指令。

```
<%@ Page EnableSessionState="False" %>
```

上述设定并不会毁坏其他页面建立的会话，而只会禁止从该页面访问 Session 对象的值。

另一种方法就是通过改变窗体页的属性来选择，在下拉列表框中选择 DOCUMENT，然后选择 enableSessionState 属性，如图 9.2 所示。

可以在 enableSessionState 属性的下拉列表框中的 True、False、ReadOnly 的选项中选择一种。

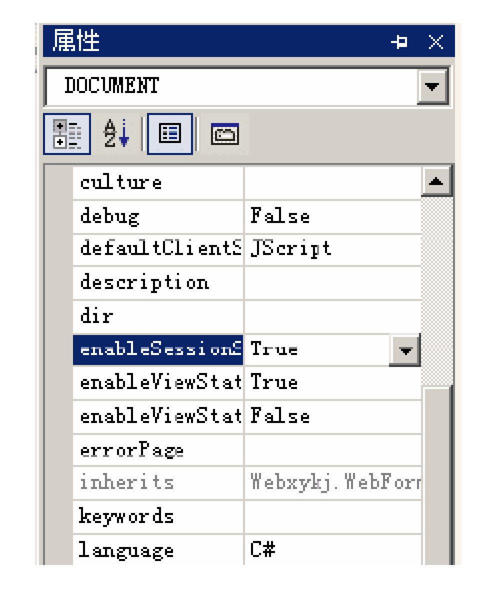


图 9.2　设置 enableSessionState

3. 管理对话

Session 对象提供了 Timeout 属性，用来设置 Session 的有效时限，以分钟为单位。默认情况下有效时限为 20 分钟。即如果在有效时间内没有链接 Web 服务器，对 Session 的设置将自动失效。可以在网页中延长或缩短 Session 的有效时间。例如语句

```
Session.Timeout=60 ;
```

就可以将 Session 的有效时间延长至 60 分钟。

如果需要终止 Session 的使用时，可以调用 Abandon()方法。语句如下。

```
Session.Abandon();
```

注意：被 Session 使用的关键字(名称)不分大小写，因此不要用大小写来区分多个变量。

9.5　Cookie 状态

和 Session 对象一样，Cookie 对象也是保存下来作为单个客户共享的状态，但是这个对象保存的位置与 Session 不同。Session 被保存在服务器端，而 Cookie 是保存在浏览器端的。在 Cookie 中只能含有较少量的信息，大多数浏览器不超过 4096B(有些新的浏览器

可以达到 8192B)。

Cookie 最早出现是在 Netscape Navigator 2.0 中。后来 ASP 也引入了这个技术，它的作用是与 Session 对象相结合来识别客户。每当客户开始连接站点时，系统将自动在内存块中创建一个客户有关的会话状态，同时创建一个客户的 ID 存放在浏览器端，与当前的客户唯一地联系起来。这样，服务器保存了 Session，浏览器保存了 Cookie(客户的 ID)。当下一次客户发出请求时，请求的客户将被要求提交客户的 ID，两者对照以正确地还原原来的会话状态。这就是在无状态协议的 HTTP 条件下保持客户标志的方法。

可以通过 Response.Cookies.Add() 方法直接向浏览器写入 Cookie，通过 Request.Cookies 方法读取已经设置好的 Cookie。

写入 Cookie 的语句是

```
Response.Write(cookie.Value.ToString());
```

读取指定的 Cookie 时的语句是

```
HttpCookie cookie = Request.Cookies["Cookie 的名称"];
```

Cookie 是保存在客户端的字符串，它会影响客户的行为，但又不受客户的直接管理。虽然它只是一种标志(数字、字符串)而不是程序，不可能用它来收集客户的信息，破坏客户的隐私，但有的客户仍然不放心，也可能是不愿意别人占用自己的空间，有些客户在浏览器中禁止使用 Cookie，这就给识别客户带来了困难。

ASP.NET 现在已经完全解决了在不使用 Cookie 的情况下，识别客户的方法。解决的方法很简单，只需要在应用程序的根目录下的 Web.config 文件中，对<sessionState>节点进行配置，其他任何程序都不需要修改。为什么一定要在应用程序的根目录下配置？因为会话状态的设置是应用程序范围的设置。站点中的网页要么全都使用该配置，要么全都不使用。配置的语句是

```
<sessionState cookieless="UseUri" />
```

或

```
<sessionState cookieless="AutoDetect" />
```

配置时，当编写到“cookieless= ”语句时，将弹出 AutoDetect、UseCookies、UseDeviceProfile、UseUri 4 种选择。选择 AutoDetect 或 UseUri 均可以在无 Cookies 的条件下识别客户。

虽然在<sessionState>节点中还可以配置会话状态管理的其他方面，包括存储介质和连接字符串等，但是，就 Cookie 而言，只需设置 cookieless 属性即可。

系统是如何在无 Cookie 的条件下识别客户的呢？原来当进行了前面的设置以后，系统将会要求客户自动将客户端的资源信息嵌入到客户调用的 URL 语句中。例如在使用 Cookie 的情况下，某客户调用网页时的 URL 是

```
http://yourserver/folder/default.aspx
```

现在设置了不使用 Cookie 的配置，调用的语句的 URL 将变成

```
http://yourserver/folder/(session ID here)/default.aspx
```

其中 session ID here 代表客户的资源信息所处的位置。该信息已经被插入到 URL 的语句中。由于客户资源信息对于客户来说具有唯一性，因此可以利用它与 Session 对象结合，一起来识别客户。

9.6　简单的应用示例

例 9.1　网页之间传送数据。

在网站的设计中经常需要从一张网页向另一张网页传送数据，或者在多张网页中共享数据。为了实现这一功能可以选择使用以下两种方法。

(1)　通过 URL 中的参数传送数据。

URL 的作用是给网页定位，在 URL 的后面增加参数也可以给新网页传递数据。利用 Url 只能传送一些简单的数据，不能传送类似对象这样的复杂数据，而且传送的数据将直接显示在浏览器的 URL 中。因此只适合于传送保密性不强的数据。例如某按钮的 Click 事件代码如下。

```
private void Button1_Click(object sender, System.EventArgs e)
{
    Response.Redirect("Webform2.aspx?
    name=Cheng&email=Chengli@yahoo.com.cn");
}
```

程序运行中若单击该按钮，将转向 Webform2.aspx 网页，同时网页上方的小窗口中将显示如下数据：

```
http://网站域名/Webform2.aspx?name=Cheng&email=Chengli@yahoo.com.cn
```

在数据中以“？”符号为分界，前一部分是网页的网址，后一部分为传送的参数。注意，当有多个参数时，各参数之间用&分割，不能有空格。

如果网页中用 TextBox1 控件放入 name 的值，用 TextBox2 控件放入 email 的值时，上述事件的语句应改写为

```
private void Button1_Click(object sender, System.EventArgs e)
{
Response.Redirect("Webform2.aspx?name="+TextBox1.Text+"&"+"email="+TextB
ox2.Text);
}
```

这种方式与使用 HTML 的表单用 get 方式传送数据时的结果相同。

在接收数据的网页中利用 Request.QueryString 方法，来获取传来的参数。假定新网页中设置了两个 Label 控件(Label1 与 Label2)。可在 Page_Load 事件中编写如下代码。

```
private void Page_Load(object sender, System.EventArgs e)
{
    Label1.Text=Request ["name"];
    Label2.Text=Request ["email"];
}
```

显示的结果如图 9.3 所示。

接收数据	
name:	**Cheng**
E_Mail	Chenli@yahoo.com.cn

图 9.3 接收并显示传来的数据

在本示例中，是将获取的参数放置在两个 Label 控件中。

(2) 利用 Session 对象在多张网页中共享数据。

利用 Session 对象可以在同一客户的多张网页之间共享数据。方法是先将需要共享的数据存入 Session 对象中，哪个网页需要该数据时，通过该 Session 的名即可调入使用。下面用代码举例说明。

先将数据存入 Session 对象中。例如：

```
private void Button1_Click(object sender, System.EventArgs e)
{
    Session["name"]=TextBox1.Text;
    Session["email"]=TextBox2.Text;
    Server.Transfer("anotherwebform.aspx");
}
```

在另一张网页中接收数据。方法是从 Session 对象中取出数据，将其赋给某些控件。例如：

```
private void Page_Load(object sender, System.EventArgs e)
{
    Label1.Text=Session["name"].ToString();
    Label2.Text=Session["email"].ToString();
}
```

例 9.2 统计并显示当前访问网站的人数。

设计的要点如下。

(1) 打开“全局应用程序类文件”(Global.asax)。利用文件中的 void Application_Start() 方法，将应用程序对象(如 Application["usercount"])的初始值设为 0。其代码如下。

```
void Application_Start(object sender, EventArgs e)
    {
       Application["usercount"] = 0;
    }
```

(2) 每当客户打开网站时，将应用程序对象加 1。代码如下。

```
void Session_Start(object sender, EventArgs e)
{
    Application.Lock();
    Application["usercount"] = (int)Application["usercount"] + 1;
    Application.UnLock();
}
```

(3) 每当客户退出网站时，将应用程序对象减 1。代码如下。

```
void Session_End(object sender, EventArgs e)
{
```

```
    Application.Lock();
    Application["usercount"] = (int)Application["usercount"] - 1;
    Application.UnLock();
}
```

注意：只有在 Web.config 文件中的 sessionstate 模式设置为 mode= InProc 时，才会引发 Session_End 事件。

(4)　利用某张网页的 Page_Load()事件显示当前访问网站的人数。代码如下。

```
protected void Page_Load(object sender, EventArgs e)
{
    Label1.Text = "当前本站有 " + Application["usercount"].ToString()+" "+
    "位客户访问";
 }
```

例 9.3　通过一个“客户登录及保护”的简单示例来说明会话对象的特点。这里的示例只是用来说明 Session 对象的作用，它还不是一个实际的应用程序。因为实际的应用中还需要用到数据库或其他方面的知识，这些知识将在后面章节中介绍。

假定有一批网页只对部分客户开放。访问者需先输入自己的姓名及密码，密码正确时才能打开被保护的页面。

登录页面的布置如图 9.4 所示。



图 9.4　客户登录的简单示例

在【确认】按钮的 Click 事件中编写如下代码。

```
private void Button1_Click(object sender, System.EventArgs e)
    {
     if(TextBox2.Text=="111111")   //假定正确的代码就是 111111
        Response.Redirect("WebForm2.aspx");
    else
      Response.Write("密码有错！");
}
```

用这些代码能不能保护其他网页呢？不能。因为因特网是个开放的系统。客户完全可以绕过对密码的验证，直接通过 URL 访问其他网页，而不必通过登录网页。这就好比一个单位建立了传达室却没有修建保护的围墙。

利用 Session 对象可以构筑一座“虚拟的围墙”。方法是在【确认】按钮的代码中增加对 Session 对象的设置，以便在其他网页中共享。具体代码如下：

```
private void Button1_Click(object sender, System.EventArgs e)
     {
      if(TextBox2.Text=="111111")
      {
         Session["Pass"]="Right";
         Response.Redirect("WebForm2.aspx");
      }
     else
     {
       Session["Pass"]=null;
       Response.Write("密码有错！");
     }
}
```

代码中增加了对 Session["Pass"]对象的设置。如果密码正确，执行语句

```
Session["Pass"]="Right";
```

如果密码不正确，执行语句

```
Session["Pass"]=null;
```

在被保护的网页的 Page_Load 事件中增加如下代码。

```
private void Page_Load(object sender, System.EventArgs e)
  {
  if(Session["Pass"]==null)
   {
    Response.Redirect("WebForm1.aspx");
   }
}
```

代码中的 WebForm1.aspx 代表登录网页。当网页打开时(Page_Load 事件发生)，先将 Pass 会话对象中的字符串取出，然后进行判断，若字符串为 null，则返回登录页面，要求重新输入密码。这就防止了不经过验证密码的客户打开网页的可能。

例 9.4 通过网上投票的示例来说明 Cookie 对象的使用。

假定下个月将举行一次“中、韩国家足球队之间的友谊赛”，现在想通过网上投票来预测一下比赛的结果。为避免“一人多投”，程序中利用了 Cookie 的功能。

网页的界面如图 9.5 所示。

图 9.5 结果预测

【提交】按钮的代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
if (Request.Cookies["vote"] == null)
   {
    // 将投票结果存入数据库或文件中(此略)
    Label1.Text = "您的投票已经收到，谢谢！";
    Response.Cookies["vote"].Value = "-1";
    Response.Cookies["vote"].Expires = DateTime.Now.AddMonths(1);
   }
 else Label1.Text = "您已经投过票了！";
}
```

代码中使用了 Cookies["vote"]对象。当客户第一次投票时(Request.Cookies["vote"] == null)将投票情况保存以便统计(保存与统计方法此处都略去)。与此同时将一个 cookie 对象作为标志暂存于投票者的浏览器中。这个标志保存的时间用语句 Response.Cookies ["vote"].Expires 来确定。在上面的语句中：

```
Response.Cookies["vote"].Expires = DateTime.Now.AddMonths(1);
```

表示保存时间从投票时算起持续一个月。这里的保存时间有多种选择。如 AddYears(n)、AddMonths(n)、AddDays(n)、AddHours(n)、AddMinutes(n)、AddSeconds(n)等。

如果在持续时间内客户再次投票，其结果将不被接收，同时提示“对不起，您已经投过票了！”

值得注意的是，用这种方法统计票数，并不是一种严格的计票方法，存在着客户造假的潜在危险。因此，此方法只适合用做一般性的舆论调查，不能用于需要严格计票的场合。

9.7　Web 窗体页的生命周期

Web 窗体页的生命周期代表一个窗体页从生成到消亡所经历的阶段，以及在各阶段中执行的方法、使用的信息、保持的数据、呈现的状态等。程序设计者掌握这些知识，会对理解和分析某些问题有所帮助。

下面按照执行的顺序简要地讲述窗体页生命周期各阶段执行的内容。

(1)　初始化：主要执行 Page 的 Init 事件和 OnInit 方法。

(2)　加载视图状态：主要执行 LoadViewState 方法，就是从 ViewState 中获取上一次的状态，并依照页面的控件树的结构，用递归来遍历整个树，将对应的状态恢复到每一个控件上。

(3)　处理回发的数据：主要执行 LoadPostData()方法，用来检查客户端发回的控件数据的状态是否发生了改变。

(4)　加载：本阶段主要是触发 Load 事件，执行 Page_Load 方法。

(5)　预呈现：预呈现这个阶段就是执行在最终呈现之前所做的状态的更改，因为在呈现一个控件之前，必须根据它的属性来产生 HTML，比如 Style 属性，这是最典型的例子。在预呈现之前，可以更改一个控件的 Style，当执行预呈现的时候，就可以把 Style 保存下来，作为呈现阶段显示 HTML 的样式信息。

(6)　保存状态：这个阶段就是把状态写入 ViewState。

(7)　呈现：将对应的 HTML 代码写入最终响应的流中。

(8)　处置：实际上就是执行 Dispose 方法，在这个阶段会释放占用的资源，例如数据库连接等。

(9)　卸载：最后，页面会执行 OnUnLoad 方法，触发 UnLoad 事件，处理在页面对象被销毁之前的最后处理，实际上 ASP.NET 提供这个事件只是设计上的考虑，通常资源的释放都会在 Dispose 方法中完成，所以这个方法也变成不怎么重要了。

9.8 小　　结

Web 应用程序与 Windows 桌面应用程序最重要的区别就在于状态管理。

HTTP 是一种不保持状态的通信协议。但是在网站中有时需要保持某些状态，此时，可以使用状态管理功能。

视图状态是保持本窗体的状态，目的是在窗体网页的往返中保持连续性。该状态由系统设置的隐含字段自动保存和恢复。

应用程序状态是全局性的状态，用于保存整个网站共享的数据，用代码来实现。代码中要注意处理可能引发的竞争问题。

会话状态与 Cookie 状态都是保存单个客户的状态，不过会话状态保存在服务器端，而 Cookie 状态保存在浏览器端。两者结合来识别客户。为了防止客户关闭 Cookie 状态，ASP.NET 2.0(3.5)解决了无 Cookie 条件下识别客户的问题。即在应用项目根目录下的 Web.config 文件中进行设置。经过设置后客户将会把资源信息嵌入到 URL 的调用语句中，由于资源信息对于客户使用的设备来说具有唯一性，因此可以将它与 Session 结合来识别客户。

网页的生命周期是指网页从生成到消亡的各个执行阶段。了解各个阶段执行的方法以及它们的先后执行顺序，对分析问题或理解、处理某些问题有所帮助。

9.9 习　　题

1. 填空题

(1) 状态分为 4 种类型，它们是________、________、________和________。

(2) 下面是设置和取出 Session 对象的代码。

设置 Session 的代码是

```
Session["greeting"] = "Hello Wang!";
```

取出该 Session 对象的语句如下。

```
string MyVar=________________;
```

(3) 下面是使用 Application 对象时防止竞争的代码。

```
Application.______;                //锁定 Application 对象
Application ["counter"] = (int)Application ["counter"] + 1;
Application.________;              //解除对 Application 对象的锁定
```

(4) 在浏览器已经封闭 Cookie 的条件下，为了识别客户应该在应用程序的根目录下的 Web.config 文件中，对<sessionState>节点做如下配置。

```
<sessionState cookieless=" ______" />
```

或

```
<sessionState cookieless="______________ "/>
```

(5) 改变 Session 的有效时间的语句是_________________________。

(6) 废除 Session 的语句是_________________________________。

2. 选择题

(1) Session 与 Cookie 状态之间的最大区别在于____________。

A. 存储的位置不同　B. 类型不同　C. 生命周期不同　D. 容量不同

(2) 默认情况下，Session 的有效时间是___________。

A. 30 秒　B. 10 分钟　C. 20 分钟　D. 30 分钟

3. 判断题

(1) HTTP 是一个不保持状态的通信协议。这就意味着当浏览器与服务器之间的会话结束，它们之间的连接也就自动断开了，下一次会话与本次连接无关，两次连接之间不存在任何联系。 (　)

(2) 使用 HTML 控件时将不能保持视图状态。 (　)

(3) 视图状态可以在各个网页之间共享。 (　)

(4) Session 对象可以在同一对话的不同网页之间共享。 (　)

4. 简答题

(1) 为什么说用 Session 对象来表示电子商务中的购货车是最佳的选择？

(2) 为什么要保持视图状态？ASP.NET 中是如何保持视图状态的？

5. 操作题

(1) 创建多个网页，在其中一个网页中输入姓名和密码，要求当转移到其他网页时，这个姓名和密码将自动传送到新的网页中并显示出来。

(2) 利用 Cookie 编写一段舆论调查程序，以避免客户重复投票。

第 10 章 数 据 验 证

程序运行时，对一些输入的数据进行验证是很有必要的，因为不正确的输入很可能会给后续的应用带来麻烦。例如，某客户发来的购货订单中，遗漏了地址，你向哪里发货？不仅如此，有些错误的输入还会给系统运行造成直接的影响，轻者会降低系统的运行效率，重者可能破坏系统的正常运行。

当然，为了验证数据可以自行编写验证代码，但这样做比较麻烦。利用系统提供的验证控件，可以很方便地验证输入数据的正确性。

本章将首先介绍各个控件的验证原则、使用方法，最后用一个综合示例作为总结。具体包括以下几个问题：

- 概述。
- 验证控件的类型。
- 各验证控件的使用方法。
- 自定义控件。
- 分组校验技术。
- 拒绝机器人行为。
- 综合示例。

10.1 概 述

验证工作最好放在客户端进行。当在客户端输入完数据，向服务器提交以前应对数据进行检测，如果发现错误，立即提示并要求改正，改正前不向服务器提交信息。这种处理方式可以将改正错误的过程放在提交以前，减少网上的无效传输。

有两个原因使得不能依赖客户端的验证：第一，由于相当一部分客户端的设备功能弱，不具备验证能力，此时验证工作只能放在服务器端进行；第二，恶意的客户能够比较容易地破坏客户端的验证脚本，或者想方设法绕过客户端的校验。

因此，从安全的角度出发，除非人为地取消了服务器端验证，不论客户端是否进行了验证，服务器端的验证都是不可缺少的。当客户向服务器提交数据之后，服务器都毫无例外地调用验证程序来逐个检查客户的输入。如果发现任何输入数据有错误时，整个页面将自行设置为无效状态，并发出错误信息。

10.2 验证控件的类型

系统提供了 5 种验证控件(包括程序设计者自行定义的控件)和一个汇总控件。各种验证控件的作用如表 10.1 所示。

表 10.1　验证控件

验证类型	验证控件名	作用说明
必需项的验证	RequiredFieldValidator	确保客户不会遗漏该项的输入
比较验证	CompareValidator	使用小于、等于、大于等比较运算符，将客户的输入与另一常量值或与另一控件的属性值进行比较
范围检查	RangeValidator	检查客户的输入是否在指定的范围内。可以检查数字、字母字符和日期的范围
模式匹配	RegularExpressionValidator	检查项与正则表达式定义的模式比较，看是否匹配。这种验证类型用于检查可预知的字符序列，如身份证号、社会保险号、电子邮件地址、电话号码、邮政编码等
自行定义	CustomValidator	使用自己编写的验证逻辑检查客户输入。这种验证类型允许检查在运行时导出的值

另外，还有一个 ValidationSummary 汇总控件，它只能与以上控件一道使用，不能单独执行验证。它的作用是将来自页面上所有控件的错误信息集中在一起进行显示。

所有的验证控件都继承于 BaseValidator 类，该类又派生于 Label 并实现了 IValidator 接口，所以所有的验证控件都继承了 Label 控件利用 span 显示的能力。另外，在 BaseValidator 类中还定义了“显示错误(ErrorMessage)”属性和“字符串颜色(ForColor)”的属性(默认情况下为红色)。如果需要，可以改变这些属性来改变错误的提示和提示的颜色。

在这些控件中，除 RequiredFieldValidator 控件以外，其他所有的控件都认为空字段是合法的。

允许将多个验证控件对某一个输入控件同时进行多方面的验证。例如可以指定某个控件既必须输入数据，同时输入的数据必须在指定的范围内。此时就可以将 RequiredValidator 和 RangeValidator 两个控件同时指向该控件。

当执行多条件验证时，验证条件之间是“逻辑与(AND)”的关系。即只有所有条件都符合要求时才能获得通过。

10.3　各验证控件的使用方法

各个控件虽然作用不同，但使用的方法却有很多共同点，因为它们都继承于共同的基类 BaseValidator。比如每个控件都有一个 ControlToValidate 属性，必须用它来指定被验证的控件。下面分别介绍各验证控件的使用方法。

10.3.1　RequiredFieldValidator 控件

RequiredFieldValidator 控件用于对一些必须输入的信息进行检验，如果一些必须输入的数据没有输入时，将提示错误。

使用这个控件的方法比较简单，将控件拖入窗体以后，关键是给它设置以下 4 个属性。

- ControlToValidate：设置被验证的控件，可以在本属性的下拉列表中选择。
- ErrorMessage：当不能通过验证时显示的错误信息。
- Display：显示错误信息的位置，包括以下 3 种选择。
  - None：不显示错误信息。
  - Static：显示在设计时控件所放置的位置。
  - Dynamic：将错误信息动态显示在页面上。
- EnableClientScript：本属性为逻辑变量，默认为 true，表示如有可能(例如浏览器版本为 Internet Explorer 4.0 以上)，先在客户端验证。若将本属性值改为 false，将不在客户端进行验证。

10.3.2 CompareValidator 控件

CompareValidator 控件用来将输入到控件(例如 TextBox 控件)的值与输入到其他控件的值或常数值进行比较。几个重要的属性的设置方法如下。

- 通过设置 ControlToValidate 属性指定被验证的输入控件。
- 如果要将输入控件与其他输入控件进行比较，将 ControlToCompare 属性设置为要与之相比较的控件。如果要将输入控件的值与某个常数值进行比较时，应将 ValueToCompare 属性设置为与之比较的常数。
- 类型(Type)属性用于设置比较数据的类型。只有在同一类型的数据之间才能够进行比较。
- 操作符(Operator)属性用来指定比较的方法，如大于、等于等。如果将 Operator 属性设置为 ValidationCompareOperator.DataTypeCheck，则 CompareValidator 控件将忽略 ControlToCompare 和 ValueToCompare 属性，并且仅仅指示输入到输入控件中的值是否可以转换为 BaseCompareValidator.Type 属性指定的数据类型。

10.3.3 RangeValidator 控件

RangeValidator 控件用于检查输入控件的值是否在指定的范围内。

RangeValidator 控件中有 4 个关键属性用来执行验证。

- ControlToValidate 属性指向被验证的输入控件。
- MinimumValue 和 MaximumValue 属性用来确定有效值范围的最大值和最小值。
- Type 属性用于设置要比较的值的数据类型。需要比较的值在比较前必须转换成表 10.2 中的类型之一。

表 10.2 数据类型

数据类型	说　明
String	字符串数据类型
Integer	32 位有符号整数数据类型

续表

数据类型	说　明
Double	双精度浮点数数据类型
Date	日期数据类型
Currency	一种可以包含货币符号的十进制数据类型

实际应用中有时输入的范围需要动态确定。下面举例说明利用程序动态确定范围的方法。

例如，利用因特网在某个度假地预订房间，要求提出预订的开始日期和结束日期，旅客只要在这个时间范围以内到达都可以，超出这个范围时，房间就不能保证。

现在用两个 TextBox 服务器控件，即 TextBoxl 和 TextBox2 记录开始日期和结束日期。而实际到达日期输入到第三个控件 TextBox3 中，该控件通过 RangeValidator 控件验证。验证的条件是 TextBox3 输入框的日期必须在前面两个日期之间。

先在窗体页界面中拖入 3 个 TextBox 控件、一个 Button 控件，并将 RangeValidator 控件拖入窗体。图 10.1 就是到达时间超出预订时间时的错误显示。

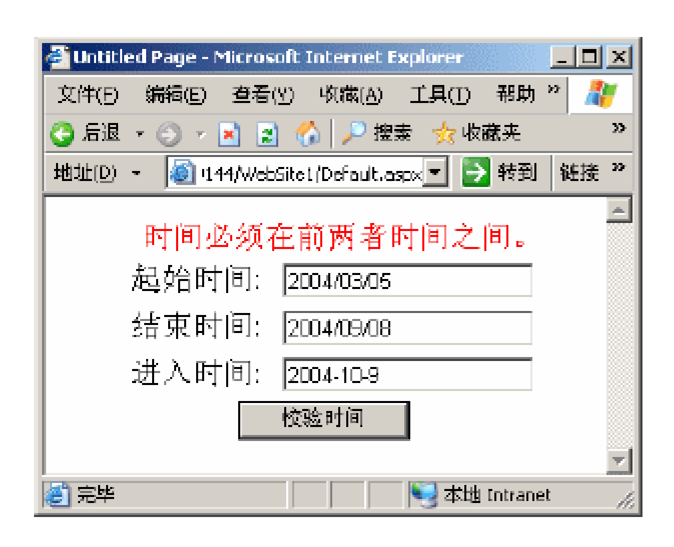


图 10.1　时间范围验证

将 RangeValidator 控件的 ControlToValidate 属性指向 TextBox3，将 Type 设为 String，将 EnableClientScript 属性设为 False。在按钮单击事件中编写如下代码。

```
void Button1_Click(object sender, EventArgs e)
  {
    DateTime tt1 = DateTime.Parse(TextBox1.Text);
    DateTime tt2 = DateTime.Parse(TextBox2.Text);
    DateTime tt3 = DateTime.Parse(TextBox3.Text);
    RangeValidator1.MinimumValue = tt1.ToShortDateString();
    RangeValidator1.MaximumValue = tt2.ToShortDateString();
    TextBox3.Text = tt3.ToShortDateString();
     RangeValidator1.Validate();
    if (!RangeValidator1.IsValid)
      {
        RangeValidator1.ErrorMessage = "时间必须在前两者时间之间。";
      }
   }
```

注意：通过编程进行验证时，应该禁用客户端脚本，以便控件不会在服务器端验证代码执行之前显示不正确的错误信息。

代码中调用了方法 Validator()进行验证，验证的结果放在属性 IsValid 中。若 RangeValidator.IsValid 为 false 时，说明不能通过验证，应该显示错误信息。

10.3.4 RegularExpressionValidator 控件

RegularExpressionValidator 控件用来验证输入的格式是否匹配某种特定的模式(正则表达式)。这类验证允许检查一些可以预知的字符序列，比如身份证号码、电子邮件地址、电话号码和邮编中的字符序列等。

除非浏览器不支持客户端验证，或者已明确禁止客户端验证(通过将 EnableClientScript 属性设置为 false)，否则将同时执行服务器端和客户端验证。

客户端的正则表达式验证实现和服务器端的略有不同。在客户端，使用的是 JScript 正则表达式语法。而在服务器端，使用的则是 System.Text.RegularExpressions.Regex 语法。由于 JScript 正则表达式语法是 System.Text.RegularExpressions.Regex 语法的子集，所以最好使用 JScript 正则表达式语法，以便在客户端和服务器端得到同样的结果。

使用本控件进行校验时，除按照前面几个控件设置属性以外，最主要的区别是将控件的 ValidationExpress 属性设置检查模式。方法是单击属性右边的省略号按钮，在弹出的对话框中选择【标准表达式】，弹出的对话框如图 10.2 所示。



图 10.2 选择正则表达式

然后选择需要检查的模式即可。

10.3.5 ValidationSummary 控件

ValidationSummary 控件用于在一个位置上集中显示来自 Web 网页上所有验证程序的错误信息。根据 DisplayMode 属性的设置，可以采用列表、项目符号列表或单个段落的形式来显示。通过设置控件的 ShowSummary 和 ShowMessageBox 属性，可以确定显示的形式。

10.4 自定义控件

使用自定义控件 CustomValidator 时，可以自行定义验证算法，并同时利用控件提供的其他功能。

为了在服务器端验证函数，先将 CustomValidator 控件拖入窗体，并将 ControlToValidate

属性指向被验证的对象，然后给该验证控件的 ServerValidate 事件提供一个验证程序，最后在 ErrorMessage 属性中填写出现错误时显示的信息。

在 ServerValidate 事件处理程序中，可以从 ServerValidateEventArgs 参数的 Value 属性中获取输入到被验证控件中的字符串。验证的结果要存储到 ServerValidateEventArgs 的属性 IsValid (true 或者 false)中。

例如，利用自定义 CustomValidator 控件验证某个输入框输入的数据能否被 3 整除。若不能被 3 整除时发出错误信息。事件处理的代码如下。

```
private void CustomValidator1_ServerValidate(object source,
   System.Web.UI.WebControls.ServerValidateEventArgs args)
    {
        int number=int.Parse(args.Value);   //取出输入的数据
        if((number % 3) == 0)               //校验能否被 3 整除
            args.IsValid=true;              //结果正确
        else
            args.IsValid=false;             //结果错误
    }
```

如果需要同时提供客户端验证程序以便让具有 DHTML 能力的浏览器先进行验证时，应该在.aspx 的 HTML 视图中用 JavaScript 语言编写验证程序，同时将验证的函数名写入控件的 ClientValidationFunction 属性中。

10.5　分组校验技术

在一个网页中通常会出现几个独立的输入部分，它们的作用不同，验证的时机也不相同，应该分别进行验证。例如，网页中既包括用于查询记录的输入部分，又包括客户认证部分，就属于这种情况。在 HTML 网页中可以设置多个表单，将这些输入控件分别放在不同的表单中，以便单独进行校验和提交数据。但是在 ASPX 网页中，每个网页就是一个运行在服务器的表单。表单的定义如下。

```
<form id = form1 runat = "server">
…
</form>
```

分组校验是 ASP.NET 2.0 版本提出来的新技术，用来解决在 ASPX 网页中多组不同输入的校验问题。它要求利用各个控件的 ValidationGroup 属性给这些控件进行分组。

现在用一个简单的示例来说明分组校验的方法。假定有两组输入控件，各包括一个输入框(TextBox)和一个按钮(Button)，并且分别放入了校验控件。为了进行分组，将其中一组的输入框、按钮和校验控件的 ValidationGroup 属性设置为 Group1，而将另一组控件的 ValidationGroup 属性设为 Group2。控件的布局如图 10.3 所示。

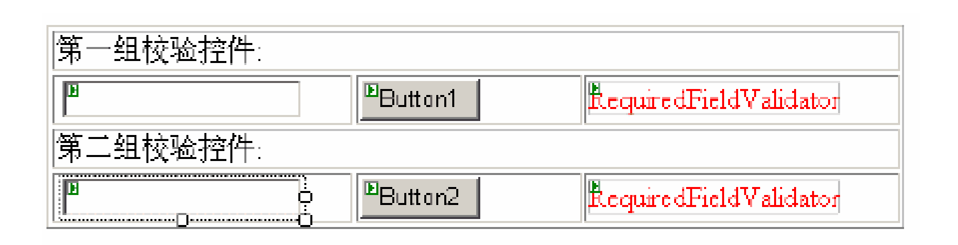


图 10.3　对验证控件分组

对应的代码如下。

```
<form id="form1" runat="server">
    <div>
     <table border="1" style="width: 680px; height: 88px">
        <tr>
        <td colspan="3">
         第一组校验控件:</td>
        </tr>
        <tr>
           <td style="width: 62px">
              <asp:TextBox ID="TextBox1" runat="server"
                ValidationGroup="Group1">
              </asp:TextBox>
           </td>
           <td style="width: 100px">
              <asp:Button ID="Button1" runat="server" Text="Button1"
                 ValidationGroup="Group1" />
           </td>
           <td style="width: 100px">
              <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
                runat="server" ErrorMessage="RequiredFieldValidator"
                ValidationGroup="Group1">
                </asp:RequiredFieldValidator
           </td>
        </tr>
        <tr>
           <td colspan="3">
              第二组校验控件:</td>
           </tr>
        <tr>
           <td style="width: 62px">
              <asp:TextBox ID="TextBox2" runat="server"
                 ValidationGroup="Group2">
              </asp:TextBox>
           </td>
           <td style="width: 100px">
              <asp:Button ID="Button2" runat="server" Text="Button2"
                  ValidationGroup="Group2" />
           </td>
           <td style="width: 100px">
              <asp:RequiredFieldValidator ID="RequiredFieldValidator2"
                 runat="server"
                 ErrorMessage="RequiredFieldValidator"
                 ValidationGroup="Group2">
                 </asp:RequiredFieldValidator>
            </td>
         </tr>
       </table>
     </div>
</form>
```

为了简化分组方法，可以一手按住 Shift 键，另一手单击组内各个控件，然后一起设置它们的 ValidationGroup 属性。

10.6　拒绝机器人行为

10.6.1　概述

在输入校验中还有一个重要的方面，就是要拒绝机器人的行为。一些黑客为了破坏网站的正常运行，常利用机器人来冒充客户进行登录。由于机器人能够在很短时间内进行大量输入，因此可以利用它来反复登录以破解原有客户的密码；或者在短时间内添加大量新客户或电子邮件客户，以造成网络堵塞，或者使服务器不堪重负等。机器人的上述行为给网站带来严重威胁。为了保证网站安全，必须拒绝机器人的行为。

1. CAPTCHA 方案简介

为了对抗机器人的非法操作，必须找到一种识别算法来区分登录的是正常人还是机器人。到目前为止，人们已经找到了几种有效的方案，其中使用得最为普遍的方案是一种被称为 CAPTCHA 的校验码方案。

注意： CAPTCHA 是美国卡内基梅隆大学研究的项目，创建于 2002 年并注册了商标，首先应用于 Yahoo!网站，现在已被广泛应用。CAPTCHA 是 Completely Automated Public Turing test to tell Computers and Humans Apart(全自动区分计算机和人类的图灵测试)的简称，是区分机器人和人类的一种程序算法。

使用这种方案时，先在登录的界面中展示一幅图片，图片中包括被扭曲的字符串以及一些其他能够影响识别的干扰符号。如果是正常人，应该能够正确识别出这段字符串。如果是机器人，按照当前机器人的智能水平，还很难在如此复杂的环境中分辨出字符串来。这样，也就分辨出究竟是人，还是机器人了。

例如，图 10.4 就是利用 CAPTCHA 算法给出的一幅典型图片。



图 10.4　CAPTCHA 示例

图片的左上方是字符串 overlooks，右上方是 inquiry。两个字符串都被扭曲，而且中间都加上了一条作为干扰的“噪音”(曲线)。

2. 几种常见的验证码图形

校验码有多种不同的形式。一般来说，算法越复杂识别越困难。图 10.5 中列出了几种常见的校验码图形。



图 10.5 几种常见的校验码图形

校验码应具备以下特点。

1) 校验码的组成

校验码一般是数字或者数字加字符组成。

2) 字符本身采用的方式

为了增加机器人识别的难度，字符本身应采用不同的方式。例如：

- 使用一些不常用的文字。
- 字体随机倾斜显示。
- 每个字随机显示不同颜色。
- 字符随机显示在不同的位置上。
- 文字采用渐变颜色，同一个字就由好几种颜色组成。

3) 背景显示

常见背景显示方案有如下几种。

- 使用干扰线。随机生成若干条干扰线，这些线的颜色跟字体的颜色相类似。
- 干扰点。背景中随机显示若干个干扰点，这些点的颜色跟字体的颜色类似。
- 干扰色块。背景中随机出现一些色块。

10.6.2 创建图形验证网页

不同的算法将产生不同的图形。下面用一个比较简单的示例来说明创建的过程。示例中将使用 4 个文件。

- 生成登录的网页。
- 类文件。
- 调用类文件的网页。

● 显示验证结果的网页。

前两个文件是主要的文件，后两个起辅助作用。其中类文件用来随机生成校验码以及干扰元素。上述 4 个文件的关系如图 10.6 所示。

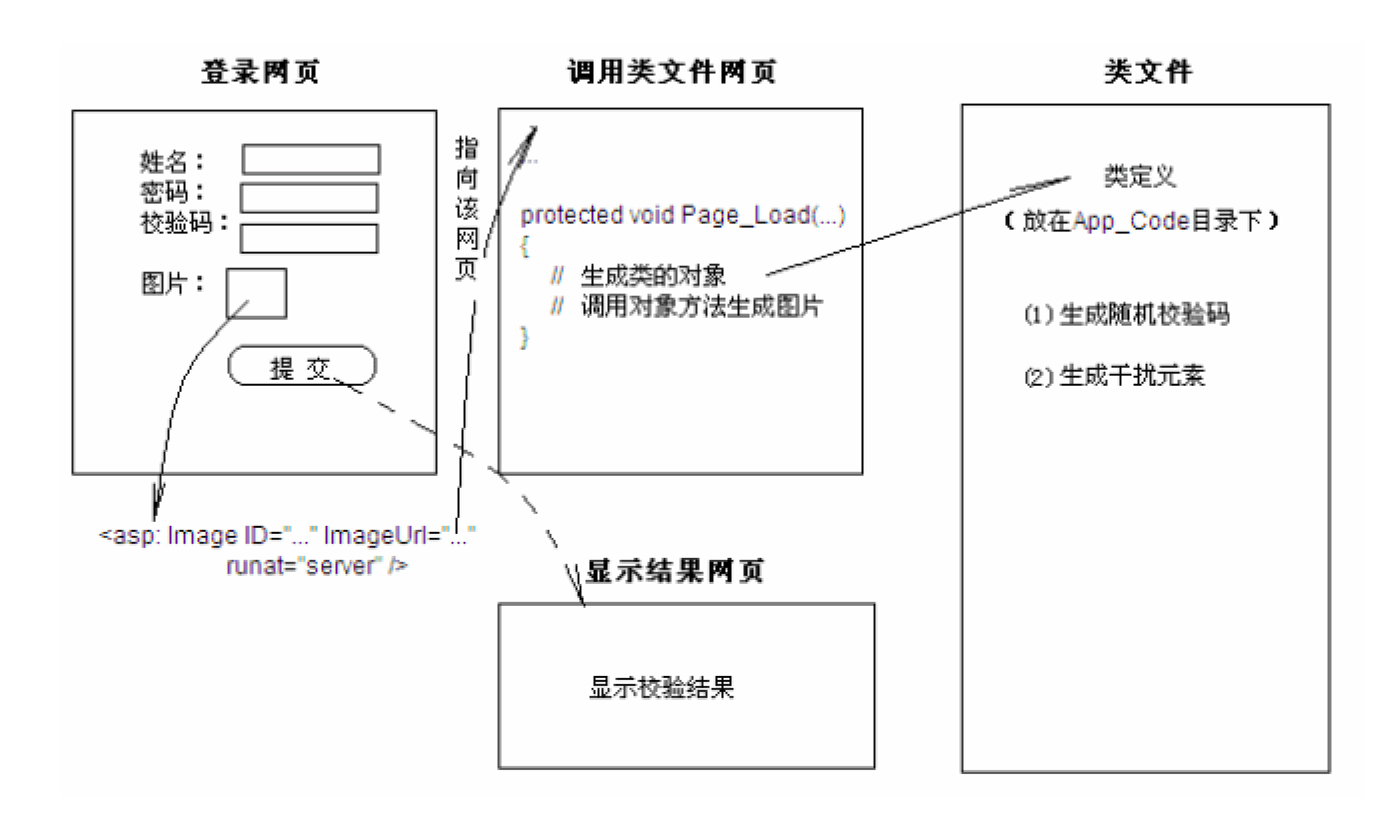


图 10.6　4 个文件的关系

在登录网页中，除放入姓名、密码输入框以外，增加一个校验码的输入框和一个图片控件，并将该图片的 ImageUrl 属性指向另一张网页(调用类文件的网页)，在该网页的 Page_Load 事件中调用类文件，以生成校验码及校验码中的干扰元素。当在登录网页中单击【提交】按钮时，将验证结果送往“显示结果网页”。

验证图片的设计，根据需要采用不同的算法。通常情况下，算法越复杂，破解也越困难。下面介绍的是一种比较简单的算法。

1. 创建登录网页

在 login.aspx 网页中设置一登录界面，如图 10.7 所示。

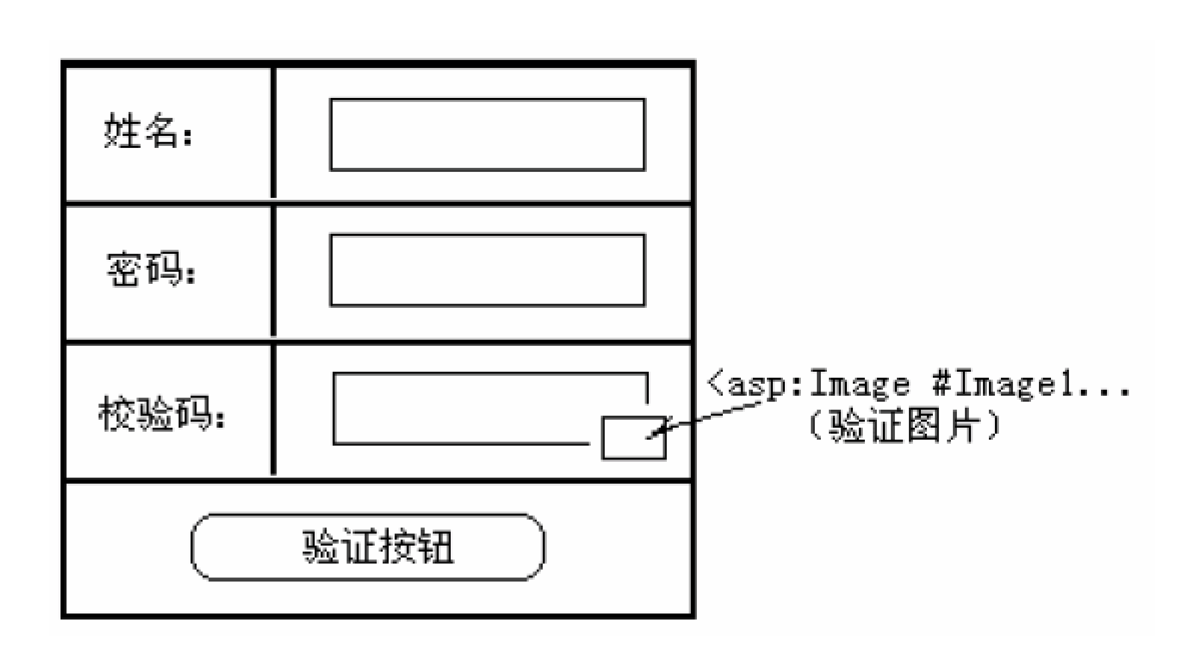


图 10.7　登录界面

这个登录界面与一般的登录界面主要不同之处在于增加了验证图片的设置。增加一个 Image 控件，并将其 ImageUrl 属性指向另一张网页。例如：

```
<asp:Image ID="Image1" ImageUrl="~/validateString.aspx" runat="server" />
```

程序运行时再通过该网页的代码生成验证图形。

2. 生成类文件

假定类名为 xt_common。类中包括两个方法。

- GenerateCheckCode()：用来生成随机校验码，并将该校验码存入 Session ["CheckCode"]中。
- CreateCheckCodeImage(string checkCode)：用来在校验码中增加干扰元素。

类的定义如下。

```
using System;
using System.Data;
using System.Configuration;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
using System.Drawing;
using System.Drawing.Imaging;
using System.Drawing.Text;
using System.IO;


/// <summary>
///xt_common 的摘要说明
/// </summary>
///
public class xt_common
{
 public xt_common()
 {
   //
   // TODO: 在此处添加构造函数逻辑
   //
  }
  public string GenerateCheckCode()
  {
   int number;
   string strCode = string.Empty;
   //随机数种子
   Random random = new Random();
   for (int i = 0; i < 4; i++) //校验码长度为 4
   {
     //随机的整数
     number = random.Next();
     //字符从 0～9,A～Z 中随机产生，对应的 ASCII 码分别为
     //48～57、65～90
     number = number % 36;
     if (number < 10)
     {
       number += 48;
     }
```

```
 else
 {
   number += 55;
 }
 strCode += ((char)number).ToString();
 }

 //在 Session 中保存校验码
 System.Web.HttpContext.Current.Session["CheckCode"] = strCode;
 return strCode;
}

/// <summary>
/// 根据校验码输出图片
/// </summary>
 public void CreateCheckCodeImage(string checkCode)
{
 //若校验码为空，则直接返回
 if (checkCode == null || checkCode.Trim() == String.Empty)
   return;
 //根据校验码的长度确定输出图片的长度
  System.Drawing.Bitmap image = new System.Drawing.Bitmap(
    (int)Math.Ceiling((double)(checkCode.Length * 15)), 20);
  //创建 Graphics 对象
  Graphics g = Graphics.FromImage(image);
  try
   {
    //生成随机数种子
    Random random = new Random();

    //清空图片背景色
    g.Clear(Color.White);

    //画图片的背景噪音线 10 条
    for (int i = 0; i < 10; i++)
    {
     //噪音线起点坐标(x1,y1),终点坐标(x2,y2)
     int x1 = random.Next(image.Width);
     int x2 = random.Next(image.Width);
     int y1 = random.Next(image.Height);
     int y2 = random.Next(image.Height);

     //用颜色画出噪音线
     g.DrawLine(new Pen(Color.Blue), x1, y1, x2, y2);
    }

     //输出图片中校验码的字体：12 号 Arial，粗斜体
     Font font = new Font("Arial", 12, (FontStyle.Bold |
     FontStyle.Italic));
     SolidBrush brush = new SolidBrush(Color.Red);
     g.DrawString(checkCode, font, brush, 2, 2);

     //画图片的前景噪音点 50 个
     for (int i = 0; i < 50; i++)
     {
```

```
            int x = random.Next(image.Width);
            int y = random.Next(image.Height);

            image.SetPixel(x, y, Color.FromArgb(random.Next()));
           }

           //画图片的边框线
           g.DrawRectangle(
            new Pen(Color.SaddleBrown),
            0, 0,
            image.Width - 1,
            image.Height - 1);
           //创建内存流用于输出图片
           using (MemoryStream ms = new MemoryStream())
           {
            //图片格式指定为 PNG
             image.Save(ms, ImageFormat.Png);

            //清除缓冲区流中的所有输出
            System.Web.HttpContext.Current.Response.ClearContent();

            //输出流的 HTTP MIME 类型设置为 image/Png
            System.Web.HttpContext.Current.Response.ContentType =
            "image/Png";

            //输出图片的二进制流

    System.Web.HttpContext.Current.Response.BinaryWrite(ms.ToArray());
      }
    }
    finally
    {
      //释放 Bitmap 对象和 Graphics 对象
      g.Dispose();
      image.Dispose();
     }
    }
      }
```

注意： ① xt_common 类文件应放置在网站的 App_Code 专用目录下。
② 在类文件中 Response、Session 都要加上前面的命名空间。

3. 生成调用类文件网页

在调用类文件的网页中通过 page_Load 事件，先生成对象，再通过对象调用方法，以生成验证图片。其代码如下。

```
public partial class validateString : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
          xt_common cca = new xt_common();
```

```
            cca.CreateCheckCodeImage(cca.GenerateCheckCode());
          }
      }
  }
```

4. 判断验证结果

在登录网页中增添验证按钮，并在它的 Click 事件中编写如下代码。

```
protected void Button1_Click(object sender, EventArgs e)
 {
  if (TextBox1.Text == "" || TextBox2.Text == "" || TextBox3.Text == "")
//检查输入框是否为空
   {
    Response.Write("客户名、密码、验证码都不能为空!");
   }
   else
    {
     string yzcode = (string) Session["CheckCode"];   //取出设定的验证码
     if (yzcode != TextBox3.Text)                //检查输入的验证码是否正确
         Response.Redirect("Result.aspx?action=验证码错误!");
            else
                Response.Redirect("Result.aspx?action=验证码正确!");
     }
}
```

代码先判断三个输入框中是否有空，如果有空时，立即显示错误。再判断输入的验证码是否有错，并将验证结果以参数的形式转到结果网页。

为了在结果网页中显示验证结果，需在网页的 Page_Load 事件中编写如下代码。

```
protected void Page_Load(object sender, EventArgs e)
    {
       Response.Write(Request["action"]);
 }
```

代码中的 action 即为传过来的参数。

10.6.3　与机器人斗争的长期性

与机器人斗争是一项长期的任务。一方面，为了网站安全必须拒绝机器人行为；另一方面，一批黑客又在不断研究利用机器人打入网站的新方法。两者之间的斗争如同病毒与反病毒一样，是一场严酷而又长期的斗争。

拒绝机器人行为的方法虽然有很多种，但是到目前为止，用得最普遍的仍然是 CAPTCHA 或者在此基础上的改进版(如 Google 开发的 reCaptcha 等)。一些大型网站(包括微软自己的网站)都经常使用这种技术。虽然这种技术并没有完全的把握来拒绝机器人行为，但却显著地提高了网站的安全等级，仍然具有重要的使用价值。

另外，我们还将在第 21 章 21.4.2 节的“对密码设置的要求”中，介绍另一种拒绝机器人行为的方法。

10.7 综合示例

下面以一个订单信息页面为例练习使用多种类型的验证控件。中间部分为输入框，是被验证的对象，右边是错误提示，通常由验证控件拖入的位置确定(最好放在被验证控件的旁边)。错误提示内容在控件的 ErrorMessage 属性中设置。程序运行时，只有错误出现时才会显示出来。

将【姓名】、【口令】、【订单号】、【地址】4 个文本框设置为必须输入验证(RequiredFieldValidator)控件，如果输入为空时提示相应的错误信息。为【重复口令】文本框设置比较验证(CompareValidator)控件用来和上一次输入的口令进行比较，看两者是否一致。为 E_mail 文本框设置模式验证框(RegularExpressionValidator)检查输入是否符合 E_mail 的格式要求。下面设有一图形验证码以拒绝机器人行为。最后设置一个汇总(ValidationSummary)控件，用来汇总验证的结果。验证界面如图 10.8 所示。

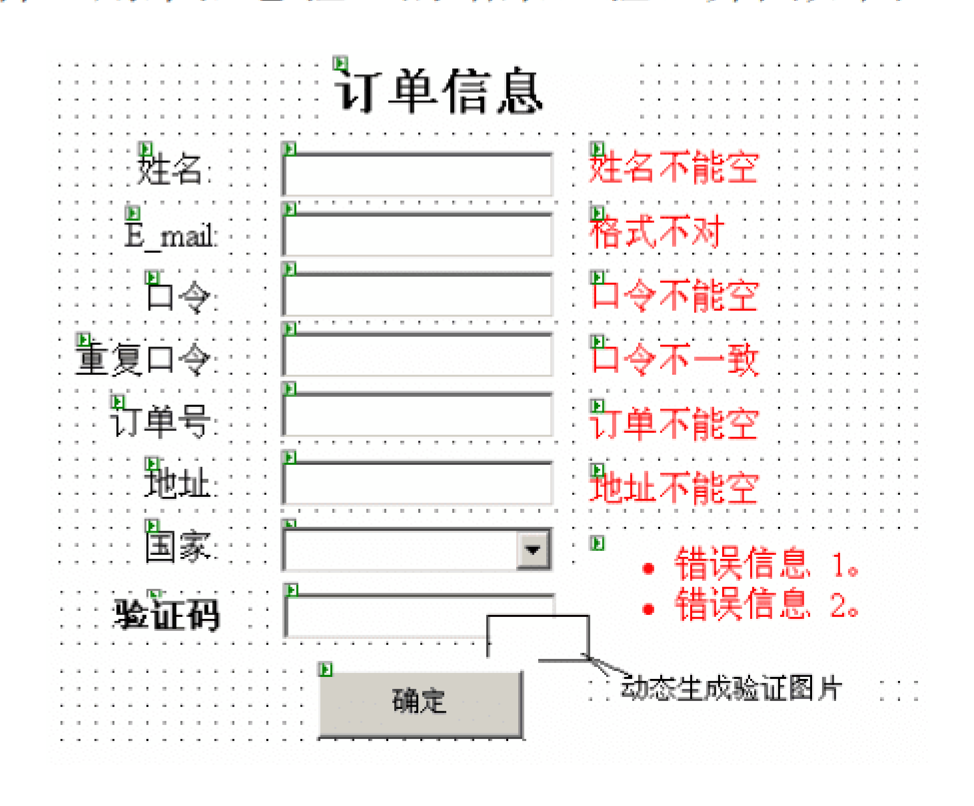


图 10.8 综合验证界面

10.8 小 结

由于网站工作在开放的环境中，可能遇到各种复杂的情况，因此对输入数据进行校验就显得特别重要。在 ASP.NET 中，校验工作都是在服务器端进行。在可能的情况下，将自动调用客户端校验作为一种补充，以减少错误信息在网络上的往返次数，提高处理的效率。

各种校验控件虽然作用不同，但是使用的方法却有很多共同点，都需要将属性指向被校验的控件，指定错误发生时提示的语句，其他属性的设置则根据控件的作用不同而有所不同。在这些控件中除 RequiredFieldValidator 外，其他控件都认为空的输入是允许的。因此有时需要将 RequiredFieldValidator 控件与其他控件一起指向输入控件时，才能避免输入错误的发生。

与黑客的斗争，以及与机器人的战争都是长期而又艰巨的任务。此消彼长，只有不断探索新技术才能应对各种不断改变的攻击手段。本章中介绍了目前应用得比较普遍的

CAPTCHA 方法，这种方法的特点是利用一种特殊的图形来区别是正常人还是机器人。

由于每个 ASPX 网页本身就是一个表单，而同一个网页中却可能存在着多组独立的验证控件，此时可以利用 ASP.NET 3.5 提供的分组校验技术将这些校验控件进行分组，以便让它们在不同的时机完成自己独立的校验工作。

10.9 习 题

1. 填空题

(1) 在设计阶段必须将各个验证控件的______________属性指向被验证的控件。

(2) 使用 RegularExpression 控件验证输入时，首先要将本控件的____________属性设置成检查的模式。

2. 选择题

(1) 现在需要验证某个 TextBox 控件的输入数据是否大于 0。此时应该使用的验证控件是________。

A. CompareValidator　　B. CompareValidator 与 RequiredFieldValidator

C. RangeValidator　　D. RangeValidator 与 RequiredFieldValidator

(2) 现在需要验证某个 TextBox 控件输入的年龄是否大于 18 且小于 65。此时应该使用的验证控件是________。

A. CompareValidator　　B. CompareValidator 与 RequiredFieldValidator

C. RangeValidator　　D. RangeValidator 与 RequiredFieldValidator

(3) ValidatorSummary 验证控件的作用是________。

A. 检查总和数　　B. 集中显示各个验证的结果

C. 判断有无超出范围　　D. 检查数值的大小

(4) 在图片校验码的图片中增加一些线条或点阵是为了________。

A. 使图片更美观　　B. 使图片更加规整

C. 干扰机器人的识别　　D. 使图像更丰富

3. 判断题

(1) ASP.NET 主要依靠在浏览器端对输入进行验证工作，因为在浏览器端验证可以将错误发现在提交之前，以减少信息的传输量。 ( )

(2) 除 RequiredFieldValidator 控件以外，其他验证控件都将被检查对象为空时认为是合法的输入。 ( )

(3) CompareValidator 控件既可以用来与某个常量比较，也可以用来与另外某个控件的输入进行比较。 ( )

4. 简答题

(1) 为什么 ASP.NET 对数据输入的验证以服务器验证为主，浏览器端验证为辅？

(2) 简述利用 CompareValidator 控件进行验证时几个属性的设置方法。包括的属性有

ControlToValidate、ControlToCompare、ValueToCompare、Type 和 Operator。

(3) 举例说明自定义控件(CustomValidator)的设计方法。

(4) 什么情况下需要进行分组验证，用什么属性对控件进行分组？

5. 操作题

(1) 设计一个密码输入的网页验证界面。输入的界面包括姓名、密码、重复密码、E_mail 等项。若输入完整、正确时，转向另一网页；若输入错误时，除分别显示以外，还需要汇总显示错误。假定密码为 111111。

(2) 利用 CustomValidator 控件设计一个自定义的验证控件。

(3) 设计一个图片校验码程序以拒绝机器人行为。

第 11 章　ADO.NET 简介

数据管理和分析常常是现代管理的核心，因此，开发企业管理网站时，访问、管理和分析数据总是程序的关键环节。然而访问和管理数据本身并不是一件简单的事情，这是由于两方面的因素形成的。首先，使用的数据可能来自数据库、文件、XML 文档等不同的数据源。以数据库为例，市场上还存在着不同类型的数据库，它们要求的接口也各不相同。其次，由于网站工作在 Internet 开放的环境中，对网络数据库的访问比单机数据库的访问需要解决更多的矛盾，例如数据安全、访问效率，以及多客户同时访问时可能引发的竞争等，都需要得到妥善的解决。

从本章开始，用连续 8 章来讲述与数据访问有关的问题，重点讲述访问网站数据库。本章介绍的 ADO.NET 数据模型以及数据源控件是后续章节的基础。本章讲述的主要问题包括:

- 从 ODBC 到 ADO 数据库的通用接口。
- ADO.NET 的数据模型。
- 数据源控件。

11.1　从 ODBC 到 ADO 数据库的通用接口

当前市场上存在着数十种不同类型的数据库，常用的有 Access、SQL Server、Oracle、Informix、DB2 等。这些数据库分别由不同的公司开发，技术都比较成熟。由于这些数据库采用的数据格式和接口各不相同，因此当应用程序访问它们时，就需要分别编写不同的接口，这种需要给应用程序的设计带来了麻烦。解决的方法就是由系统提供不同数据库的驱动程序，然后放在应用程序与数据库之间作为中间环节。

微软公司提供的通用接口，多年来已经经历了几次大的改进：ODBC→OLEDB→ADO→ADO.NET。

11.1.1　ODBC 通用接口

ODBC(Open Database Connectivity，开放数据库互联)是一种用 C 语言编写的，由多种函数组成的应用程序接口(Application Program Interface，API)。这些接口将数据库底层的操作隐藏在 ODBC 的各种驱动程序之中。应用程序只需要用统一的接口指向 ODBC，然后再由 ODBC 调用不同的驱动程序来驱动不同类型的数据库。ODBC 接口如图 11.1 所示。

尽管通过 ODBC 已经能够驱动大多数常用的数据库，但因为这种编程接口还过于复杂，而且还没有进行优化。所以后来微软又在 ODBC 的基础上专门针对 Access 库(*.mdb)创建了优化编程接口 DAO(Data Access Object，数据访问对象)；针对 SQL Server 数据库，创建了优化编程接口 RDO(Remote Data Objects，远程数据对象)，这种接口还能应用

于 Oracle 数据库。

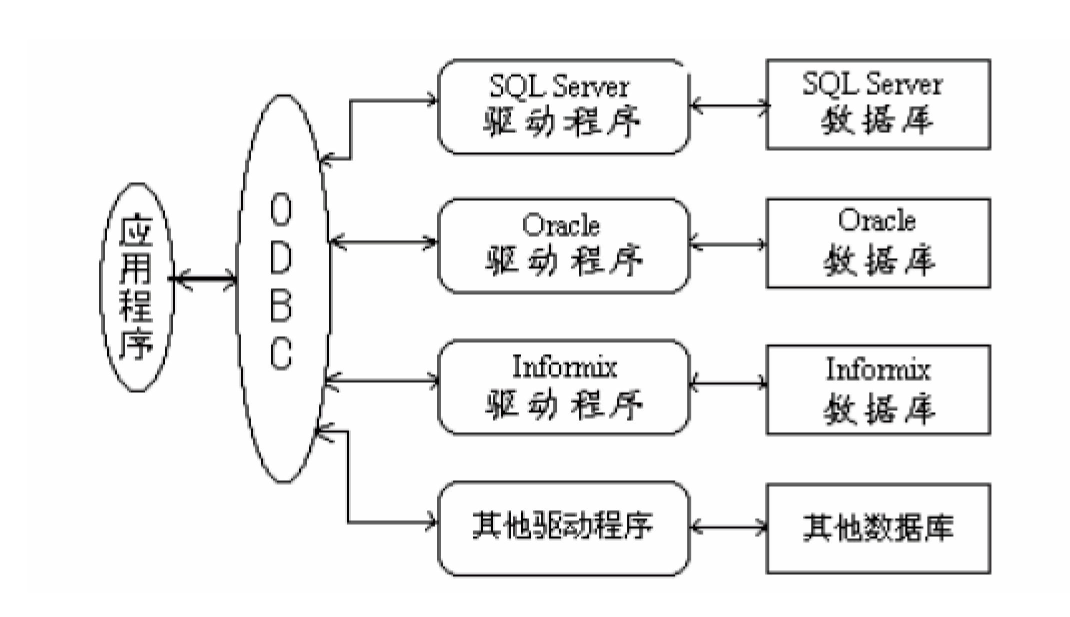


图 11.1　ODBC 接口示意

11.1.2　ADO 通用接口

ADO(ActiveX Data Object，动态数据对象)是微软进军 Internet 后的产品，它是建立在 OLE DB 技术基础上的接口技术。OLE DB 在 ODBC 的基础上，用面向对象的思想对 ODBC 的函数重新进行了分类和包装，形成了一种新的标准。可以说 ODBC 是 OLE DB 的子类，而 OLE DB 是 ODBC 的基类。利用 OLE DB 不仅能访问关系型数据库，还能访问非关系型数据(如文件等)。

ADO 又对 OLE DB 的接口进行了优化。ADO 是 ODBC 和 OLE DB 的上层接口技术。它比 RDO、DAO 等接口具有更高的性能、更小的容量及更简便的操作。

使用 ADO 接口几乎能够访问全部常用的数据库，如 Access、SQL Server、Oracle、Informix 等，还能访问非关系型文件。

ADO、OLE DB 和 ODBC 三者之间的关系如图 11.2 所示。

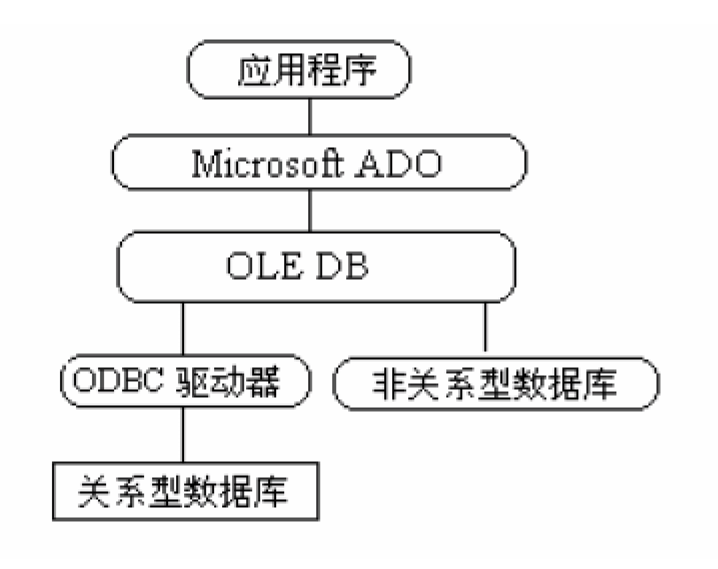


图 11.2　ADO、OLE DB 和 ODBC 三者之间的关系

11.2　ADO.NET 的数据模型

ASP.NET 使用 ADO.NET 数据模型。该模型从 ADO 发展而来，但它不只是对 ADO 的改进，而是采用了一种全新的技术。主要表现在以下几个方面。

- ADO.NET 不是采用 ActiveX 技术，而是与.NET 框架紧密结合的产物。
- ADO.NET 包含对 XML 标准的完全支持，这对于跨平台交换数据具有十分重要

的意义。

- ADO.NET 既能在与数据源连接的环境下工作，又能在断开与数据源连接的条件下工作。特别是后者，非常适合于网络应用的需要。因为在网络环境下，保持与数据源连接，不符合网站的要求，不仅效率低，付出的代价高，而且常常会引发由于多个客户同时访问时带来的冲突。因此，ADO.NET 系统集中主要精力用于解决在断开与数据源连接的条件下数据处理的问题。

11.2.1　数据访问的层次结构

ADO.NET 访问数据采用层次结构，其逻辑关系如图 11.3 所示。

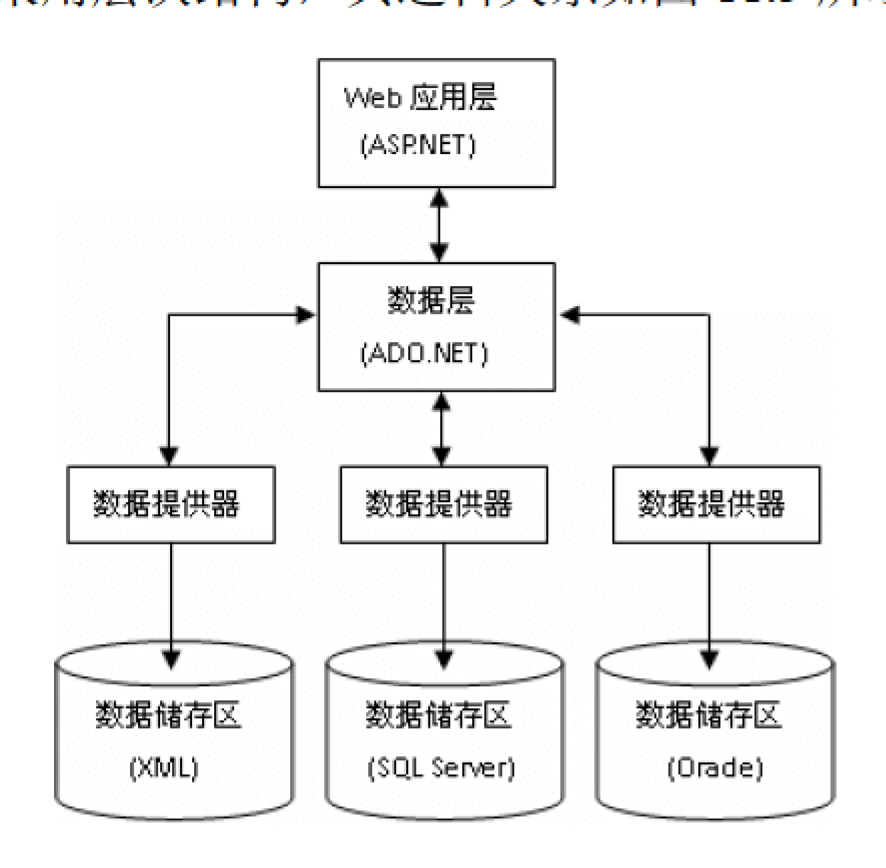


图 11.3　ADO.NET 的层次结构

图的顶层代表网站，底层代表各种不同类型的数据源，包括不同类型的数据库、XML 文档等。中间是数据层(Data Layer)，下面是数据提供器(Provider)。在这个层次结构中，数据提供器起到了关键的作用。

Provider 相当于 ADO.NET 的通用接口。不同的数据提供器对应于不同类型的数据源。每个数据提供器(Provider)相当于一个容器，包括一组类以及相关的命令，它是数据源与数据集(DataSet)之间的桥梁。它可以根据需要将相关的数据读入内存中的数据集，也可以将数据集中的数据返回到数据源。

11.2.2　数据集与数据提供器

在 ADO.NET 中数据集与数据提供器是两个非常重要而又相互关联的核心组件，数据集(DataSet)与数据提供器(Provider)的关系如图 11.4 所示。

图的左边代表数据集(DataSet)，右边代表数据提供器(Provider)。

数据集是实现 ADO.NET 断开式连接的核心，从数据源读取的数据先缓存到数据集中，然后被程序或控件调用。数据源可以是数据库或者 XML 数据。

数据提供器用于建立数据源与数据集之间的联系，它能连接各种类型的数据，并能按要求将数据源中的数据提供给数据集，或者从数据集向数据源返回编辑后的数据。

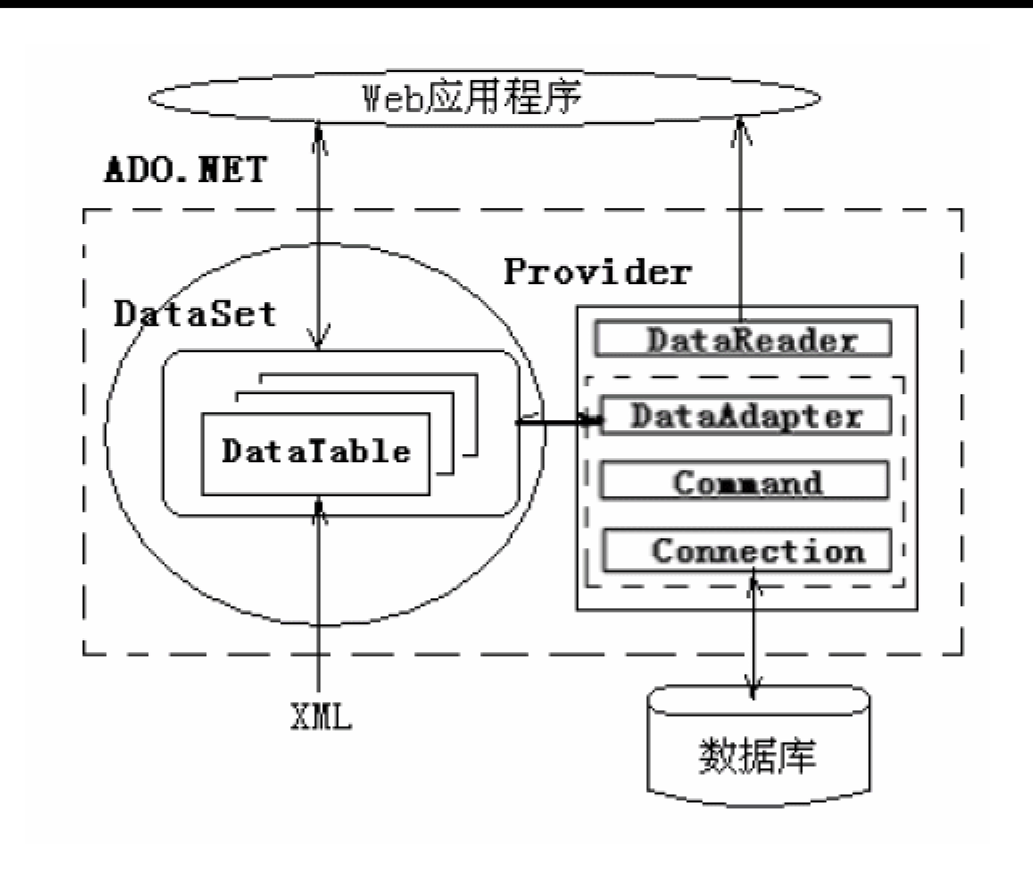


图 11.4　数据集与数据提供器

1. 数据集(DataSet)

数据集相当于内存中暂存的数据库，不仅可以包括多张数据表，还可以包括数据表之间的关系和约束。允许将不同类型的数据表复制到同一个数据集中(其中某些数据表的数据类型可能需要做一些调整)，甚至还允许将数据表与 XML 文档组合到一起协同操作。

数据集从数据源中获取数据以后就断开了与数据源之间的连接。允许在数据集中定义数据约束和表关系，增添、删除和编辑记录，还可以对数据集中的数据进行查询、统计等。当完成了各项数据操作以后，还可以将数据集中的数据送回数据源。

数据集的这些特点为满足多层分布式应用的需要跨进了一大步。因为编辑和检索数据都是一些比较繁重的工作，需要跟踪列模式、存储关系数据模型等。如果在连接数据源的条件下完成这些工作，不仅会使总体性能下降，还会影响到可扩展性的问题。

创建数据集对象的语句是

```
DataSet ds = new DataSet ();
```

或者

```
DataSet ds = new DataSet ("表名");
```

语句中 ds 代表数据集对象。前一条语句是先建立一个空数据集，以后再将已经建立的数据表包括进来；后一条语句是先建立数据表，然后建立包括该数据表的数据集。

在数据集中包括以下 4 种子类。

1) 数据表(DataTable)

数据表用来存储数据。一个数据集可以包含多张表，每张表又可包含多个行和列。数据表的创建有两种方式：第一，当将数据加载到数据集时，会自动创建一些表；第二，以编程方式创建 DataTable 的对象，然后将这个对象添加到 DataSet 的 Tables 集合中。

提取数据集中的数据表的语句是

```
DataTable dt = ds.数据表名;
```

其中，dt 代表数据表对象；ds 代表数据集对象。

2)　数据行(DataRow)

数据行是给定数据表中的一行数据，或者说是数据表中的一条记录。它可能代表一个学生、一位客户、一张订单或者一件货物的相关数据。DataRow 对象的方法提供了对表中数据的插入、删除、更新和查看等功能。提取数据表中的行的语句如下。

```
DataRow dr = dt.Rows[n];
```

其中，DataRow 代表数据行类；dr 是数据行对象；dt 代表数据表对象；n 代表行的序号(序号从 0 开始)。

3)　数据列(DataColumn)

数据表中的数据列(又称字段)定义了表的数据结构，例如可以用它确定列中的数据类型和大小，还可以对其他属性进行设置。例如，确定列中的数据是否只读的、是否主键、是否允许空值等；还可以让列在一个初始值的基础上自动增值，增值的步长还可以自行定义。

获取某列的值需要在数据行的基础上进行。语句如下。

```
string dc = dr.Columns["字段名"].ToString();
```

或者

```
string dc = dr.Column[index].ToString();
```

两条语句具有同样的作用。其中 dr 代表引用的数据行，dc 是该行某列的值(用字符串表示)，index 代表列(字段)对应的索引值(列的索引值从 0 开始)。

综合前面的语句，若想取出数据表(dt)中第三条记录中的“姓名”字段，并将该字段的值放入一输入框(textBox1)中时，语句可以写为

```
DataTable dt = ds.Customers                // 从数据集中提取数据表
DataRow dRow = dt.Rows[2 ];               // 从数据表提取行
string textBox1.Text=dRow["CompanyName"].ToString();  // 从行中取出字段的值
```

语句执行的结果是：从 Customers 数据表的第三条记录中，取出字段名为 CompanyName 的值，并赋给 textBox1.Text。

4)　关系(DataRelation)

表之间的关系由相关的列定义。在关系型数据库中，关系是指两个表之间外键约束的组合。为了将一张表与另一张表联系起来，可以简单地创建一个 DataRelation，它将指出一张表中的哪一列与另一张表中的哪一列相联系。

2. 数据提供器(Provider)

Provider 作为数据集与数据源之间的桥梁，它相当于一个容器，包括 4 种核心类，其类名及其作用如下。

- Connection(连接)类：用于建立与数据源的连接。
- Command(命令)类：用于设置适合于数据源的操作命令，以便执行检索、编辑或输出参数等数据操作。
- DataAdapter(数据适配器)类：每张表对应一个数据适配器，用来向数据集中填入数据，或者从数据集中读出数据。

- DataReader(数据读取)类：用于从数据源向应用程序读取只向前的、只读的、无缓冲的字符流。

下面进一步说明上述类的作用。

1) Connection 类

Connection 类提供了对数据源连接的封装。类中包含连接方法以及描述当前连接状态的属性。在 Connection 类中最重要的属性是 ConnectionString。该属性用来指定服务器名称、数据源信息以及其他登录信息。以数据库的连接对象为例，类名为 SqlConnection。其创建的语句是

```
SqlConnection sqlConnection1 = new SqlConnection();
```

设置 ConnectionString 属性的语句是

```
this.sqlConnection1.set_ConnectionString (
"workstation id=\"CHJ-IQDTYJ8FHYC\";  //服务器名
user id=cheng;password=cheng;         //安全信息
initial catalog=Northwind;persist security info=False");
// 数据库名以其他参数
```

2) Command 类

Command 类是对数据源操作命令的封装。对于数据库来说，这些命令既可以是内联的 SQL 语句，也可以是数据库的存储过程。

由 Command 类生成的对象只能在连接的基础上，对连接的数据源指定相应的操作。例如：

```
SqlCommand command1 =
new SqlCommand("SELECT * FROM Employees", sqlConnection1);
```

语句将生成一命令对象 command1，对由 sqlConnection1 连接的数据源指定检索(SELECT)操作。

3) DataAdapter 类

数据适配器(DataAdapter)利用连接对象(Connection)连接的数据源，使用命令对象(Command)规定的操作从数据源中检索出数据送往数据集，或者将数据集中经过编辑后的数据送回数据源。

数据适配器将数据填入数据集时调用方法 Fill()。语句如下：

```
dataAdapter1.Fill (dataSet10.Products);
```

或者

```
dataAdapter1.Fill (dataSet11, "Products");
```

其中，dataAdapter1 代表数据适配器名；dataSet11 代表数据集名；Products 代表数据表名。

当 dataAdapter1 调用 Fill() 方法时将使用与之相关联的命令组件所指定的 SELECT 语句从数据源中检索行。然后将行中的数据添加到 DataSet 中的 DataTable 对象中，如果 DataTable 对象不存在，则自动创建该对象。

当执行上述 SELECT 语句时，与数据库的连接必须有效，但不需要用语句将连接对

象打开。如果调用 Fill()方法之前与数据库的连接已经关闭，则将自动打开它以检索数据，执行完毕后再自动将其关闭。如果调用 Fill ()方法之前连接对象已经打开，则检索后继续保持打开状态。

一个数据集中可以放置多张数据表，但是每个数据适配器只能对应于一张数据表。

4) DataReader 类

使用 DataReader 类可以实现对特定数据源中的数据进行高速、只读、只向前的数据访问。与数据集不同，DataReader 是一个依赖于连接的对象。就是说，它只能在与数据源保持连接的状态下工作。因此执行的过程往往是：

(1) 打开与数据源的连接。
(2) 调用 DataReader 类的 Read()方法。
(3) 关闭与数据源的连接。

11.3　数据源控件

11.3.1　概述

ASP.NET 3.5 在 ADO.NET 的数据模型的基础上做了进一步的封装和抽象，提供了数据源控件(DataSource Control)。数据源控件既代表数据源，又代表与数据源相连接的数据提供器和数据集。在数据源控件中还隐含有大量的、常用的基层代码。数据源控件是一个功能强大的控件。在程序运行时，这个控件虽然不会显示在界面上，但是在幕后它却能完成很多有用的工作。

使用数据源控件之前需要进行配置。在智能向导(Wizard)的指引下，数据源控件的配置很容易完成。当配置完成以后，系统内部已经根据确定的数据源自动生成了连接对象、命令对象、数据适配器对象以及数据集，并且已经调用了数据适配器的 Fill()方法，将检索出来的数据放入数据集中。

通常情况下，只要设计人员对数据源控件的属性进行适当的设置，即可完成对数据表的分页、排序、更新、删除、增添数据等项工作，而不需要手工增添其他代码。

数据源控件可以连接不同类型的数据源，如数据库、XML 文档、其他对象等，但它留给设计者的接口却非常相似。设计人员只需采用相同或相似的方法处理数据，而不必关心数据源属于什么类型。

11.3.2　数据源控件的类型

数据源控件有 7 种类型，分别用于访问数据库、平面文件、各种对象以及 XML 文件等。它们是 AccessDataSource 数据源控件、SqlDataSource 数据源控件、ObjectDataSource 数据源控件、XMLDataSource 数据源控件、SiteMapDataSource 数据源控件、LinqDataSource 数据源控件和 EntityDataSource 数据源控件。

1. AccessDataSource 数据源控件

Microsoft Access 数据库是微软提供的小型数据库。这种数据库的特点是功能比较简单，使用比较容易。如果要使用这种数据库，可以利用此种数据源控件对数据表执行选

择、插入、编辑和删除数据表记录的操作。AccessDataSource 数据源控件的全名是 System.Web.UI.WebControls.AccessDataSource。

2. SqlDataSource 数据源控件

SQL Server 数据库是微软提供的功能强大、运行可靠的数据库，ASP.NET 结合使用这种数据库是最佳的选择。选择使用这种数据库时应该使用 SqlDataSource 数据源控件。此控件还能够用来访问 Oracle、ODBC、OLE DB 等大型数据库，并对这些数据库执行选择、插入、编辑和删除操作。这是使用得最为普遍的一种数据源控件。

3. ObjectDataSource 数据源控件

ObjectDataSource 数据源控件的全名是 System.UI.WebControls.ObjectDataSource。当应用系统比较复杂，需要构建三层分布式架构时，可以将中间层的商务逻辑封装到这个控件中，以便在应用程序中共享。通过这个控件可以连接和处理数据库、数据集、DataReader 或任意其他对象。

4. XMLDataSource 数据源控件

XML 文件通常用来描述层次型数据。因此显示层次型数据的控件(例如 TreeView)可以通过 XMLDataSource 数据源控件访问和处理 XML 文件。

例如，先将 XMLDataSource 数据源控件连接到一 XML 文件。代码如下。

```
<asp:XmlDataSource
    ID="XmlDataSource1"
    Runat="server"
    DataFile="~/xml/fruits.xml">
</asp:XmlDataSource>
```

其中 fruits.xml 是一个 XML 文件。再将 TreeView 控件与 XMLDataSource 数据源控件进行数据绑定。代码如下。

```
<asp:TreeView
    ID="TreeView1"
    Runat="server"
    DataSourceID="XmlDataSource1"
    ShowLines="True">
</asp:TreeView>
```

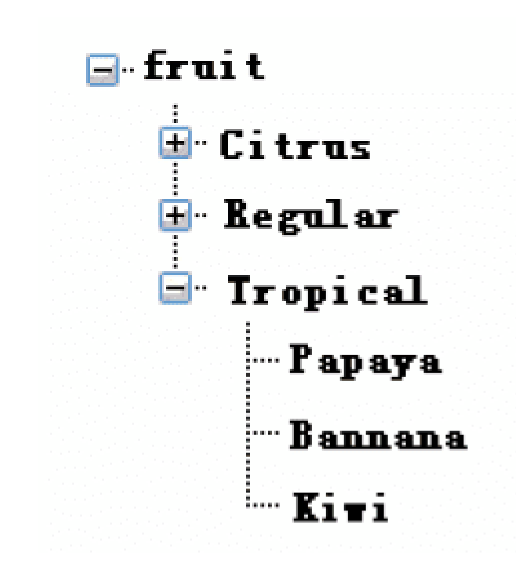


图 11.5　TreeView 的显示界面

在 TreeView 中显示的结果如图 11.5 所示。

5. SiteMapDataSource 数据源控件

控件 System.Web.UI.WebControls.SiteMapDataSource 允许用来浏览网站。方法是先建立一个网站地图文件。网站地图文件是一个 XML 文件，用来设置网页之间的逻辑关系。例如一个网站地图文件 app.sitemap 代码如下。

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap>
   <siteMapNode title="Northwind" description="Northwind"
```

```
    url="root.aspx">
      <siteMapNode title="Products"
         description="Product Line" url="Products.aspx">
            <siteMapNode title="Beverages"
              description="Tasty Beverages"
              url="Beverages.aspx" />
      </siteMapNode>
  </siteMapNode>
</siteMap>
```

然后利用 TreeView 控件通过 SiteMapDataSource 控件与网站地图文件连接，即可显示出网站中网页之间的逻辑结构。具体方法将在后面章节中讲述。

6. LinqDataSource 和 EntityDataSource 数据源控件

LinqDataSource 和 EntityDataSource 数据源控件是 ASP.NET 3.5 新提供的两个数据源控件，通过它们可以使用 LINQ 来访问不同类型的数据源对象。关于 LINQ 技术将在第 18 章中讲述。

11.4 小　　结

为了简化对网络数据库的访问，系统提供了通用接口。应用程序只需要用通用的方法连接到这些通用接口，再由通用接口分别连接到不同的数据源。这样，应用程序的设计者不必知道数据源属于什么类型，使用什么样的代码格式。微软公司提供的通用接口经历了几个重要的发展阶段，从 ODBC 发展到 OLE、ADO 再到 ADO.NET。

ADO.NET 与.NET 框架紧密结合，用类的封装取代了函数和过程，对 ADO 技术进行了全面的优化。数据集与数据提供器是 ADO.NET 技术的核心。

在 ASP.NET 2.0 与 ASP.NET 3.5 的版本中，新增加了数据源控件，它对 ADO.NET 的数据模式进行了进一步封装和抽象。数据源控件既代表数据源，又代表与数据源相连接的数据提供器和数据集。在数据源控件中还提供了其他一些服务(后面章节中讲述)，并将大量的基层代码隐藏在内部，从而大大简化了访问数据的设计过程。

ASP.NET 3.5 中包括 7 种数据源控件，分别用于连接数据库、数据对象和层次型数据，它们是 AccessDataSource、SqlDataSource、ObjectDataSource、XMLDataSource、SiteMapDataSource、LinqDataSource 和 EntityDataSource 数据源控件。

11.5 习　　题

1. 填空题

(1) 创建数据集的语句是

```
DataSet ds = ________________;
```

或者

```
DataSet ds =____________________;
```

(2) 数据提供器包括 4 种核心类，它们是_______________、________________、_______________和_______________。

2. 选择题

(1) 系统提供数据库通用接口的目的是为了________。

A. 提高程序运行的效率　　B. 应用程序设计不必考虑数据库的类型

C. 保证程序安全　　D. 易于维护

(2) XMLDataSource 与 SiteMapDataSource 数据源控件能够用来访问________。

A. 关系型数据　　B. 层次型数据

C. 字符串数据　　D. 数值型数据

3. 判断题

(1) ADO.NET 只是 ADO 的简单升级。 (　)

(2) 在数据集中可以包括多张数据表。 (　)

(3) 数据集能够在断开与数据源连接的情况下工作。 (　)

(4) DataReader 能够在断开与数据源连接的情况下工作。 (　)

(5) 数据提供器是数据集与数据源联系的中间环节。 (　)

(6) SqlDataSource 数据源控件只能用于访问 SQL Server 数据库。 (　)

(7) AccessDataSource 数据源控件只能用于访问 Microsoft Access 数据库。 (　)

4. 简答题

(1) 简述访问数据库的通用接口从 ODBC、OLE 到 ADO 再到 ADO.NET 的发展过程。

(2) 简述 ADO.NET 与 ADO 的主要不同点。

(3) ASP.NET 3.5 的数据源控件起什么作用？

(4) ASP.NET 3.5 的数据源控件有几种类型？各用于访问什么类型的数据？

第 12 章　利用 GridView 控件显示数据

从 ASP 开始，网页上的控件都是通过数据绑定的方法显示数据。在 ASP.NET 3.5 中，GridView 是功能最强的显示控件，它通过数据源控件与数据集进行数据绑定。本书将用主要的篇幅来介绍这个控件的使用方法。

本章将要讲述的问题包括：

- 数据绑定的基本概念。
- SQL Server 2005(2008) Express Edition 简介。
- 连接数据库。
- 对数据表进行分页、排序和选择。
- 利用模板美化显示。
- 显示记录中的图像。

12.1　数据绑定的基本概念

在初期的网站中，数据源与数据显示之间没有建立起自动的联系，从数据源向数据显示传递数据需要用程序设置。就是说，需要程序设计者编写一段代码，去查看数据源中的数据，判断它们是否有变化。如果有变化，再用另一段代码将变化了的数据发送到显示控件中去。

从 ASP 开始到 ASP.NET 都采用数据绑定(Data Binding)技术。数据绑定是一项非常简单而有效的技术。它将显示控件的某个属性与数据源绑定在一起。每当数据源中的数据发生变化且重新启动网页时，被绑定对象中的属性将随数据源而改变。

在 ASP.NET 中，数据绑定应用的范围非常广泛。数据集、数组、集合或者 XML 文档甚至一般的变量都可以作为数据源，大多数控件的属性都可以成为被绑定的对象。在 ASP.NET 3.5 的类库中，所有的绑定控件都从 BaseDataBoundControl 基类继承，因此都具有很多相似的功能。然后，不同类型的绑定控件继承于不同的子类。类的层次关系如图 12.1 所示。图中几个基类的情况如下。

- BaseDataBoundControl 类：是所有绑定控件的基类。
- DataBoundControl 类：是所有常用控件的基类。
- ListControl 类：是所有列表控件的基类。
- CompositeDataBoundControl 类：是比较复杂的表格控件的基类。
- HierarchicalDataBoundControl 类：是所有层次控件的基类。

本章以及后面几章中将要介绍的 GridView 控件、FormView 控件和 DetailsView 控件都从 CompositeDataBoundControl 类继承，因此它们的属性和操作方法具有很多共同点。

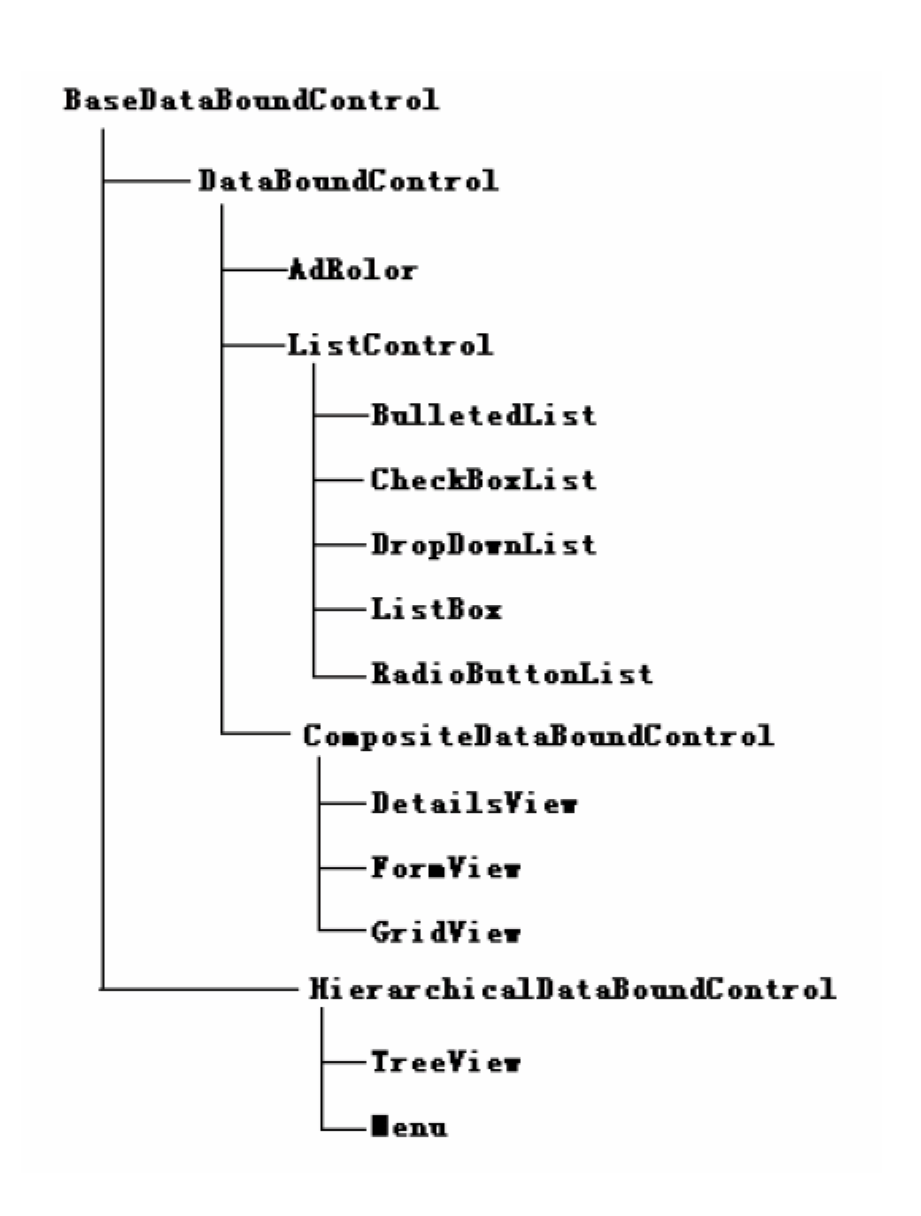


图 12.1　数据绑定控件的层次结构

12.2　SQL Server 2005(2008) Express Edition 简介

Microsoft 公司经过多年精心研制和开发，多次发布试用版来反复征求意见并进行修改补充，终于推出了 SQL Server 2005 与 2008 新一代数据库管理系统。这是公司继 SQL Server 2000 之后数据库管理系统的又一次飞跃。新产品提供了完整的编程模型，包括有 Transact-SQL、存储过程、视图、触发器、与.NET Framework 集成和使用 XML 数据类型等。不仅功能强大，而且操作相对简单，可以帮助客户轻而易举地建立、管理和维护数据库。

为满足各种不同类型的客户要求，目前微软公司已经推出了 4 种不同规模的版本，它们是 SQL Server 2008 企业版(Enterprise Edition)、SQL Server 2008 标准版(Standard Edition)、SQL Server 2008 工作组版(Workgroup Edition)和 SQL Server 2008 精简版(Express Edition)。其中企业版的功能最强，要求系统的配置也最高。SQL Server 2008 Express Edition 精简版可以看成 SQL Server 2008 的剪切版本，但与其他几个 SQL Server 2008 版本一样，使用了同样可靠的、高性能的数据库引擎，同样的数据访问 API(如 ADO.NET、SQL Native Client 和 T-SQL 等)。和其他版本相比，精简版只是规模较小，缺乏对某些企业版的功能支持。

12.2.1　SQL Server 2005(2008) Express Edition 的主要特点

Express Edition 版本的主要特点如下。

- SQL Server 2008 Express Edition 是一个允许无偿获取、随时应用并免费再分发的轻量级数据库管理系统，不仅功能强，而且易学、易用，同时系统配置的要求也

相对比较低，非常有利于中小型企业的开发应用。当需要时，还可以无缝地升级到更高一级的 SQL Server 版本。

- 与 ASP.NET 紧密集成，对动态网站提供了强有力的支持。默认情况下，安装 Visual Studio 2008 时，SQL Sever 2008 Express Edition 与 ASP.NET 3.5 一道安装，但也可以单独进行下载安装。允许直接在网站中创建数据库，也可以利用 XCopy 方法将数据库复制到网站中来，将网站和数据库接合为一体，使得在创建或访问数据库时不需要在多个工具界面之间来回切换，从而大大提高了效率，非常有利于快速开发和网站的迁移与部署。
- 在 ASP.NET 3.5 的客户认证和个性化服务中，SQL Sever 2008 Express Edition 不仅能自动生成多张数据表，还能与应用程序配合，自动保存相关数据，完成应用程序所需要的一些操作，从而大大简化了设计过程。
- 微软公司后来又发布了免费、易用的图形管理工具 SSMSE，可用来方便地管理和使用 SQL 2008 Express Edition。

注意：SQL Server 2008 Express Edition 的系统配置要求如下。

- 操作系统：Windows 2000 Service Pack 4、Windows Server 2003 Service Pack 1、Windows XP Service Pack 2。
- 单个 CPU：具有 Intel Pentium III 600MHz(或同等性能的兼容处理器)或速度更快的处理器(建议使用 1GHz 或速度更快的处理器)的计算机。
- 内存：最低 192MB 的 RAM(建议使用 512MB 或更高的 RAM)。
- 硬盘可用空间：525MB 以上。

12.2.2　在网站中创建 Express Edition 数据库

下面通过一个简单的示例来介绍在网站中创建数据库的方法。

1. 创建数据库的步骤

创建数据库的步骤如下。

(1)　右击网站名，在弹出的快捷菜单中选择【添加新项】命令，然后在弹出的对话框中选择【SQL 数据库】。在为数据库取名以后，单击【添加】按钮，系统提示“是否将数据库放置在 App_Data 的文件夹中？”，单击【是】按钮。因为只有当数据库放在该目录下时，数据库才能自动成为网站的共享文件。

(2)　如果数据库创建成功，打开服务器资源管理器，可以看到新创建的数据库名。

2. 创建数据表

创建数据表的步骤如下。

(1)　在数据库资源管理器中，展开数据库目录，右击其中的【表】项，然后在弹出的快捷菜单中选择【添加新表】命令，如图 12.2 所示。

(2)　在打开的对话框中，要求给表定义各字段(列)的属性。这些属性包括列名、数据类型和是否允许空三项，如图 12.3 所示。

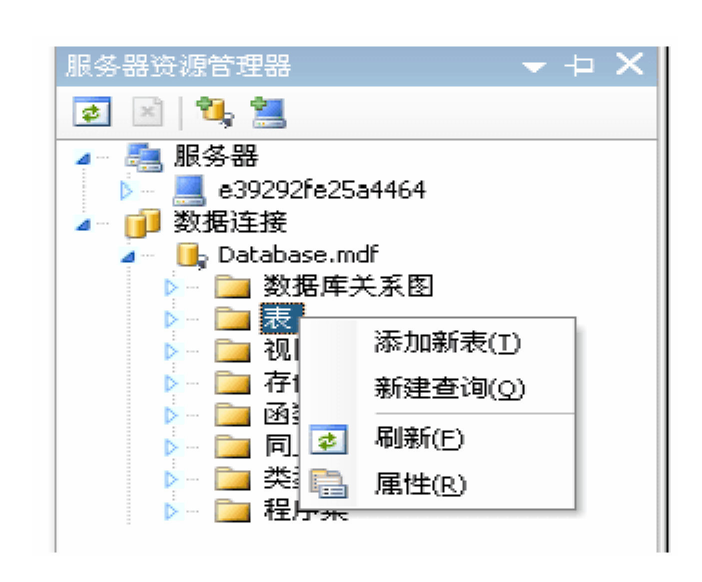


图 12.2 添加数据表



图 12.3 定义字段(列)

数据库提供的基本数据类型及其存储容量如表 12.1 所示。

表 12.1 系统提供的基本数据类型

数据类型	SQL Server 数据类型	字节数/B
字符型(Character)	char[(n)] varchar[(n)] text	0～8000 0～2G
Unicode 字符型 (Unicode Character)	nchar[(n)] nvarchar[(n)] ntext	0～8000 0～4000 0～2G
整型(Integer)	int bigint smallint tinyint	4 8 2 1
二进制(Binary)	binary[(n)] varbinary[(n)]	1～8000 1～8000
日期与时间(Date&Time)	datetime smalldatetime	8 4
精确数(Exact Numeric)	decimal[(p[,s])] numeric[(p[,s])]	2～17
近似数 (Approximate Numeric)	float[(n)] real	8 4
货币型(Monetary)	money smallmoney	8 4
图形(Image)	image	0～2G
特殊型(Special)	bit cursor timestamp sysname table sql_variant	1 0～8 8 256 0～8016

续表

数据类型	SQL Server 数据类型	字节数/B
全局标识符 (Global Identifier)	uniqueidentifer	
自定义型 (User-defined)	客户自己定义	

在上述类型中，字符串分为变长和定长两种类型。

- 变长字符串包括 nvarchar 与 varchar 两种类型。在这些类型中定义的长度，是字符串的最大长度。这两种类型虽然都可以存储变长的字符串，但两者的编码方式不同。nvarchar 采用 unicode 编码，而 varchar 采用字符编码，因此一个 nvarchar 要占用两个 varchar 的空间。但是 nvarchar 可以适用于任何语言。varchar 的优点则是占用空间少，运行速度快，但对于有的语言，数据可能会出现乱码。
- 定长字符串包括 nchar 与 char 两种类型，两者都只能存储固定长度的字符。如果输入的字符长度超过了定义的长度，将被自动切断；长度不够时则自动以空格补充。两种类型的编码方式与以上类型相似，即 nchar 用 unicode 编码，char 使用字符编码。

在本示例中，定义了 5 个字段。它们分别如下。

- bh(编号)：类型设为 int。将其设为主键(关键字)。为了使 bh 字段的值能够自动增加以避免重复，可以在该字段定义的下面设置如下属性。
  - 将【(是标识)】属性设为【是】。
  - 将【标识增量】设为 1，代表每增加一条新记录时自动加 1。
  - 将【标识种子】设为 2211，代表记录的编号从 2211 开始。

情况如图 12.4 所示。

⊟ 标识规范	是
(是标识)	是
标识增量	1
标识种子	2211

图 12.4　将 bh 设成标识

- xm(姓名)：将 xm 字段设置成 nvarchar(12)字符串类型，代表允许最长使用 12 个字符(6 个汉字)。
- xb(性别)：类型设为定长 nvarchar(4)。
- age(年龄)：类型设为 smallint(短整型)。
- phone(联系电话)：类型设为 nvarchar(30)，代表最长可以设置 30 个字符。

在上述 5 个字段中，由于 bh 是关键字不能为空。其他 4 个字段均允许空。

(3) 关闭定义对话框时，给数据表取名并存储。

(4) 右击数据表名，在弹出的快捷菜单中选择【显示表数据】。然后在弹出的对话框中输入若干条记录，如图 12.5 所示。

	bh	xm	xb	age	phone
▶	2211	成工	男	22	0731-55566611
	2212	刘归	女	20	0732-44667733
	2213	刘伟大	女	34	0643-88776655
	2214	李历历	女	23	0731-77665544
	2215	李打油	男	33	0731-44553366

图 12.5　输入数据表记录

12.3　连接数据库

下面讲述通过 GridView 控件连接数据库的步骤。

(1) 将 GridView 控件放入一新窗体页中。单击右上方的◂图标，在打开的窗口中选择【<新建数据源>】，如图 12.6 所示。

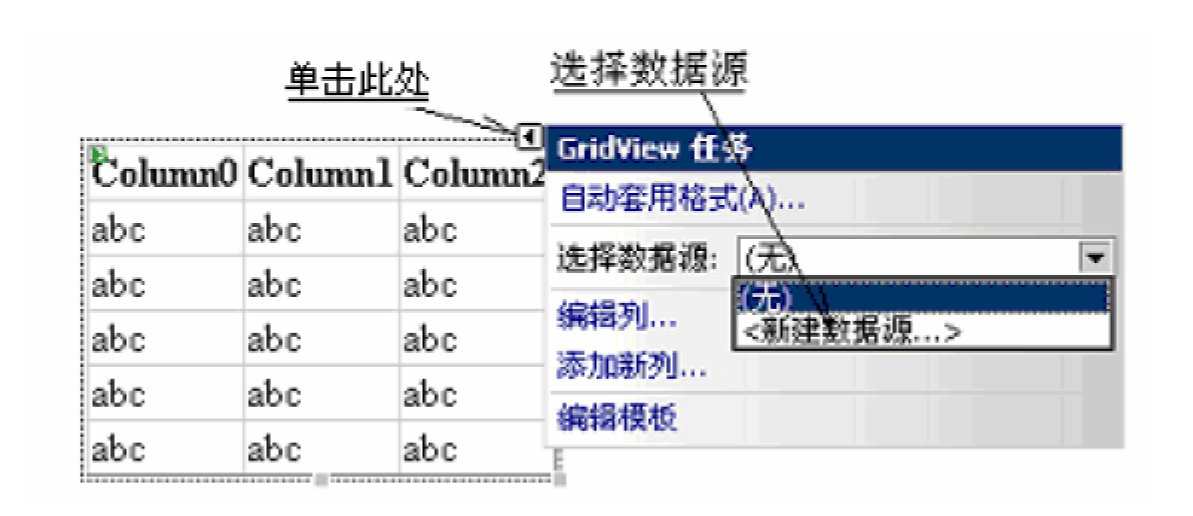


图 12.6　选择 GridView 控件的数据源

(2) 在打开的对话框中，选择【数据库】图标，然后单击【确定】按钮，如图 12.7 所示。

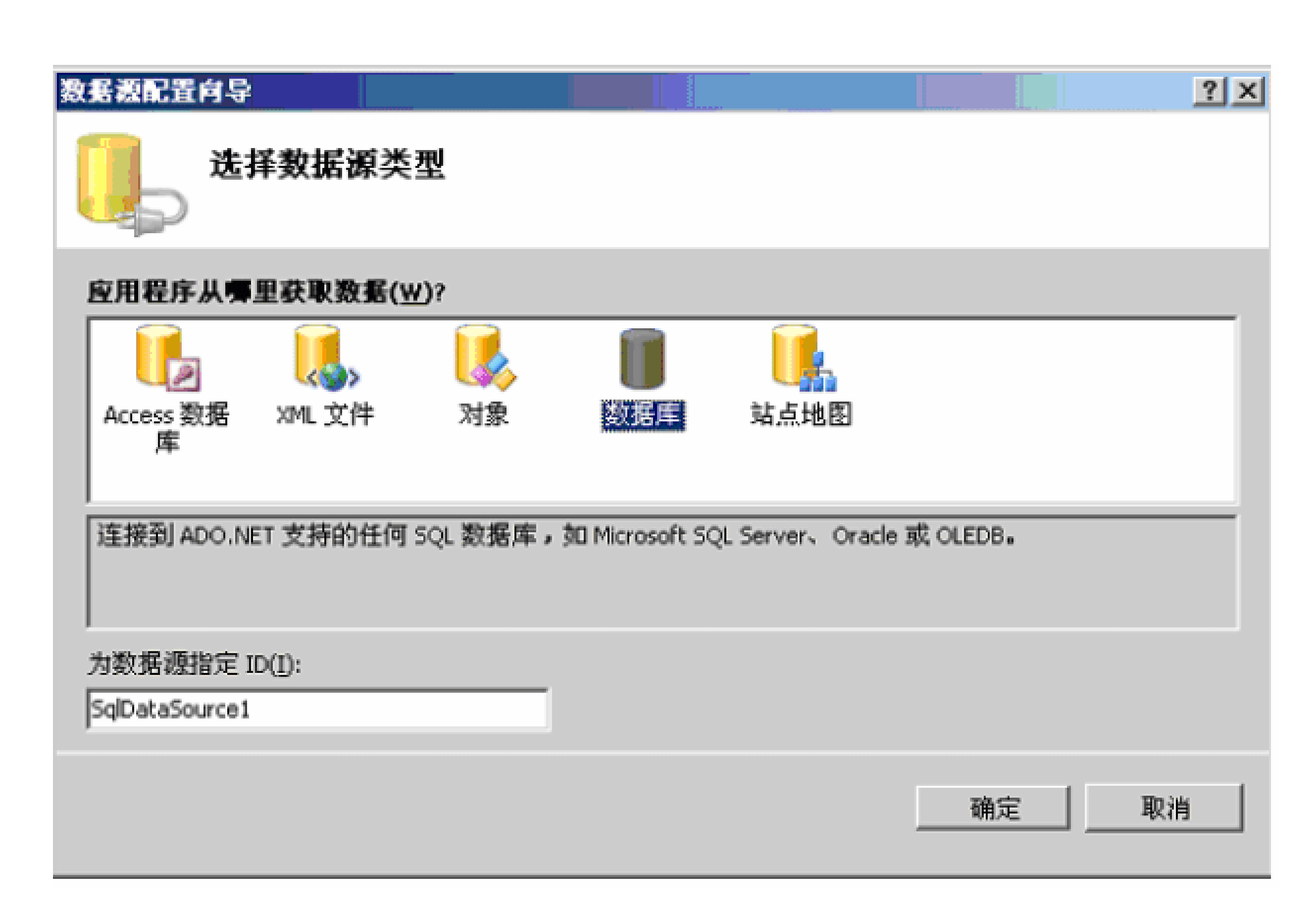


图 12.7　选择 SqlDataSource 作为数据源

(3) 在连接对话框中设置相关参数。

连接对话框情况如图 12.8 所示。



图 12.8　数据连接对话框

在下拉列表框中，可能会看到若干连接字符串名。这是前几次连接数据库时保留下来的。选用某个名字，就代表重用该次的连接。

如果是初次连接，而且连接的数据库已经放置在网站的 App_Data 目录下时，将会看到该数据库的名字，单击该数据库名即建立了与该数据库的连接。后续连接时再使用连接字符串的名字。

其他情况则需单击【新建连接】按钮，以打开【添加连接】对话框，如图 12.9 所示。

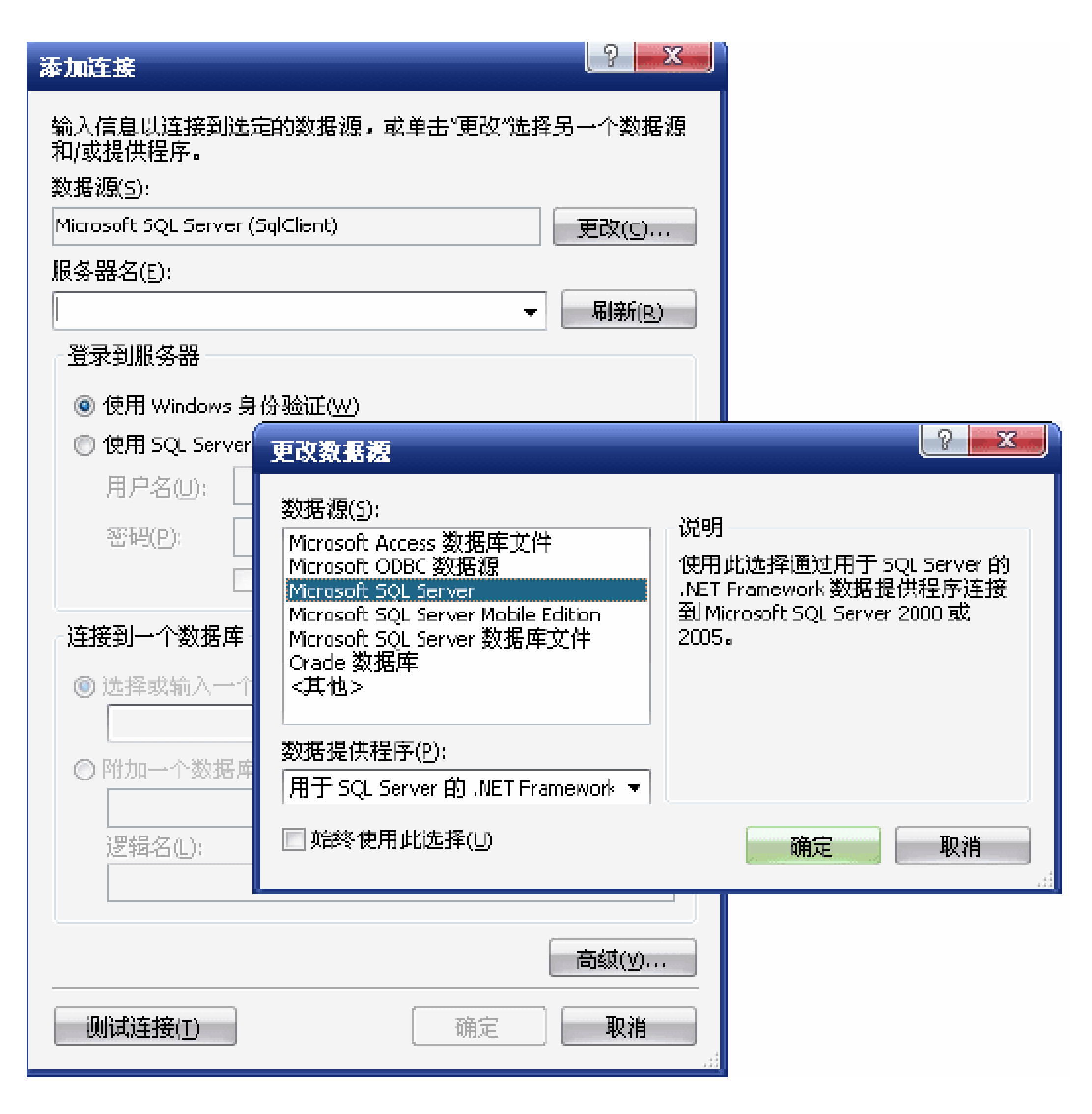


图 12.9　【添加连接】对话框

在【添加连接】对话框中完成以下设置。

① 选择数据源。

为了选择数据源，单击【更改】按钮，弹出【更改数据源】对话框。对话框中列出了多项数据源的选择。这些选择包括如下几项。

- Microsoft Access 数据库文件：用来连接 Microsoft Access 数据库文件。
- Microsoft ODBC 数据源：通过 ODBC 连接到 ODBC 驱动程序。
- Microsoft SQL Server：连接到 Microsoft SQL Server 2000、2005 或 2008。

- Microsoft SQL Server Mobile Edition：连接到移动设备(如 PDA 或 Smartphone)上的 Microsoft SQL Server 2008 Edition。
- Microsoft SQL Server 数据库文件：将数据库文件附加到本地 Microsoft SQL Server 的实例(包括 Microsoft SQL Express)上。
- Oracle 数据库：连接到 Oracle 7.3、8i 或 9i 数据库。
- 其他。

例如需要连接到 Microsoft SQL Server 2000、2005 或 2008 数据库时，应该选择 Microsoft SQL Server 选项。

② 在【服务器名】下拉列表框中填写服务器名。

连接到 SQL Server 2000 与连接到 SQL Server 2005 或 2008 数据库时，服务器名的设置稍有不同。

当需要连接到 SQL Server 2000 数据库时，服务器名就是主机名，或者用“.”(句号)代替主机名。

当需要连接到 SQL Server 2005 或 2008 Express Edition 数据库时，服务器名应设置成“主机名\SQLExpress”或“.\SQLExpress”(即句号后面加\SQLExpress)。

③ 选择连接的数据库。

当正确完成前面的设置以后，在数据库小窗口的下拉列表中就可以看见多个数据库的名字。选择其中之一即建立了与该数据库的连接。

④ 单击【测试连接】按钮，以检查连接是否有效。如果有效时再单击【确定】按钮。

(4) 系统将提示“是否将连接保存到应用程序配置文件中”。保存的目的是为了后续连接时重复使用。此时只要选中下面的复选框，即可将连接字符串保存在 Web.config 文件中。例如下面就是将连接字符串(connectionString)保存的一个结果。

```
  <connectionStrings>
  <add name="ConnectionString" connectionString="Data Source=
    .\SQLEXPRESS;AttachDbFilename=|DataDirectory|\Northwind.mdf;
       Integrated Security=True;User Instance=True"
       providerName="System.Data.SqlClient" />
</connectionStrings>
```

该连接字符串被取名为 ConnectionString。

(5) 选择数据表及其字段。

假定前面选择的数据库是 Northwind.mdf 样板库，对话框如图 12.10 所示。在下拉列表框中选择数据表，在下面的列表框中选择相应的字段。然后单击【下一步】按钮。图中选择的是 Products 数据表，并选择了其中的 5 个字段。

(6) 在新打开的对话框中单击【测试查询】按钮，看显示的结果是否符合要求，如图 12.11 所示。

如果对某个数字类型的字段限制小数位的长度时，可以在【编辑列】对话框中，将该列的 DataRormatString 属性填入“{0:F2}”字串，其中 F2 代表小数留两位。

如果显示的情况符合设计要求，单击【完成】按钮就完成了数据库连接的全部操作。

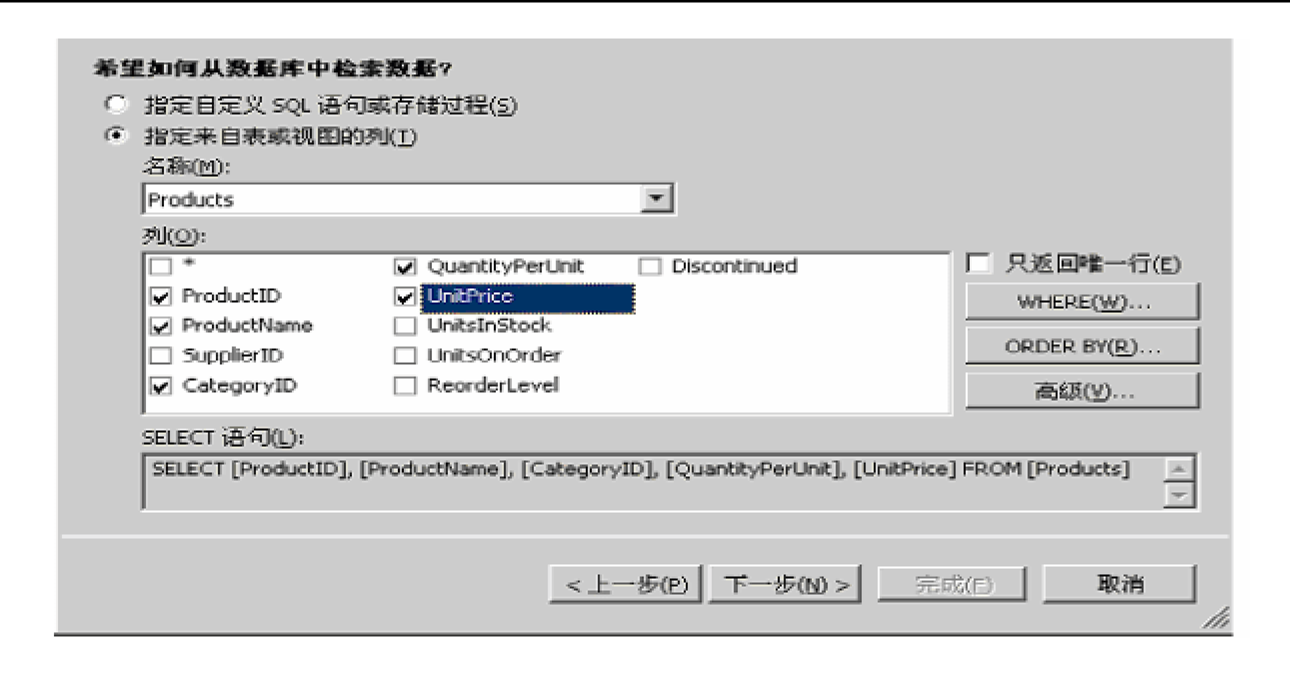


图 12.10　选择数据表以及表中的字段

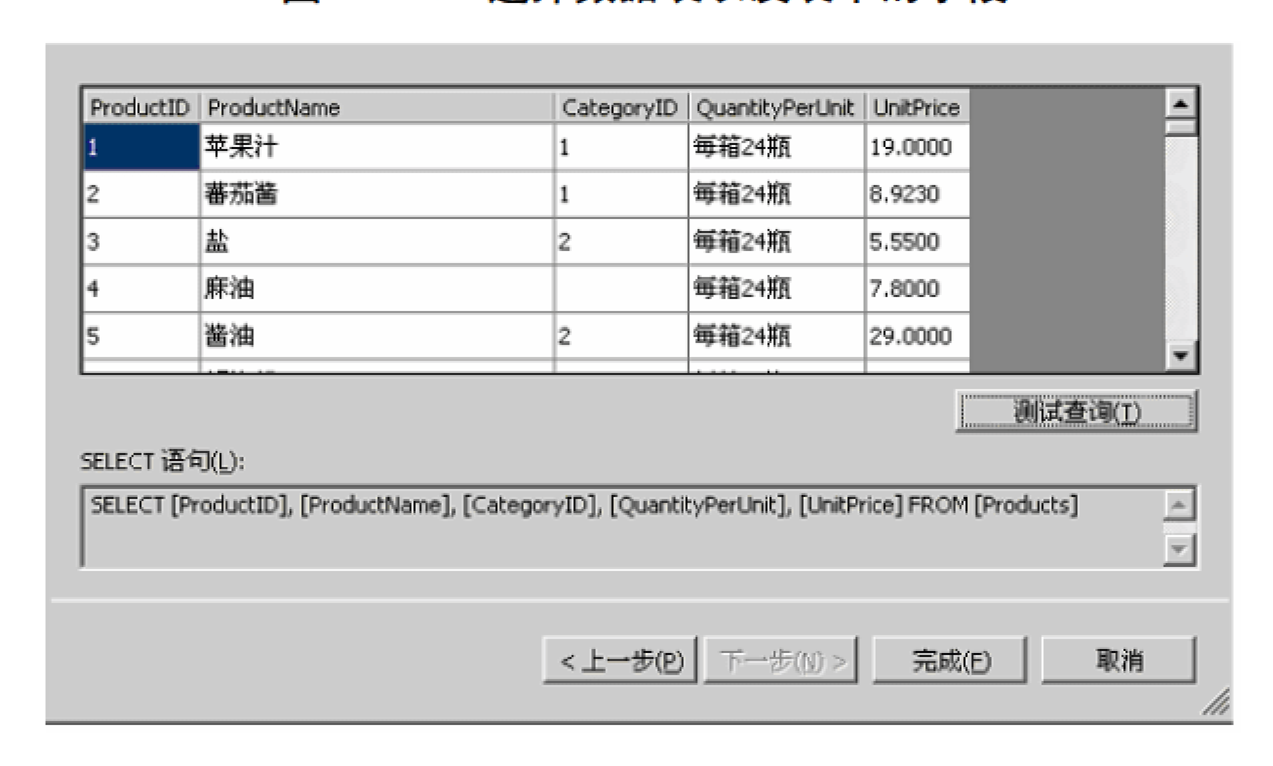


图 12.11　测试设置结果

12.4　对数据表进行分页、排序和选择

只要将 GridView 控件的 AllowPaging 和 AllowSorting 属性设置为 true(默认情况下，两个属性都为 true)，就允许对 GridView 控件中的数据表进行分页和排序。设置分页和排序可以在 GridView 控件的【属性】对话框中进行，也可以直接在与控件相连的【GridView 任务】对话框中设置。后者的设置如图 12.12 所示。



图 12.12　选择分页、排序和选择项

12.4.1 分页

选中【启用分页】复选框后还需要在 GridView 的 PageSize 属性中设置每页包含的记录数(默认的设置为每页 10 条记录)。

改变 PagerSetting 和 PagerStyles 属性，可以改变页号的显示方法。显示方法包括如下几种。

- NextPrevious：用图标显示前页、后页。
- NextPreviousFirstLast：用图标显示第一页和最后一页。
- Numeric：用数字显示页号。
- NumericFirstLast：用数字显示第一页和最后一页。

通常用“<<”表示第一页，用“>>”表示最后一页。

12.4.2 排序

选中【启用排序】复选框时，数据表各列的标题将自动转换为链接指针。程序运行时，单击某列的标题，整个表格内的数据都将以该列数据为依据从小到大地进行排列，如果想改变排序的方法，例如从由小到大，改变为由大到小或者相反时，只需再单击栏名即可完成排序方法的切换。

12.4.3 选择

选中【启用选定内容】复选框的目的是，当选择某条记录时，该记录出现与其他记录不同的显示(例如底色不同等)。为了实现这一功能，除在这里选中复选框以外，还需给 GridView 的 SelectedRowStyle 属性作出相应的设置。

经过连接、分页、排序和选择的设置后，系统的部分代码如下。

```
<!--GridView 控件的属性设置-->
<asp:GridView ID="GridView1"
    Runat="server"DataSourceID="SqlDataSource1"DataKeyNames="ProductID"
    AutoGenerateColumns="False" AllowPaging="True"
    AllowSorting="True" PageSize="5">

<!--GridView 中字段属性设置-->
      <Columns>
        <asp:CommandField ShowSelectButton="True"></asp:CommandField>
        <asp:BoundField ReadOnly="True" HeaderText="ProductID"
             InsertVisible="False" DataField="ProductID" SortExpression=
             "ProductID">
        </asp:BoundField>
        <asp:BoundField HeaderText="ProductName" DataField="ProductName"
             SortExpression="ProductName">
        </asp:BoundField>
        <asp:BoundField HeaderText="CategoryID" DataField="CategoryID"
             SortExpression="CategoryID">
        </asp:BoundField>
```

```
            <asp:BoundField HeaderText="QuantityPerUnit"
                 ataField="QuantityPerUnit"
                 SortExpression="QuantityPerUnit">
            </asp:BoundField>
            <asp:BoundField HeaderText="UnitPrice" DataField="UnitPrice"
                 SortExpression="UnitPrice">
            </asp:BoundField>
         </Columns>
</asp:GridView>

<!--数据源控件的设置-->
<asp:SqlDataSource ID="SqlDataSource1" Runat="server"
     SelectCommand="SELECT [ProductID], [ProductName], [CategoryID],
     [QuantityPerUnit],
     [UnitPrice] FROM [Products]"
     ConnectionString="<%$ ConnectionStrings:AppConnectionString1 %>">
</asp:SqlDataSource>
```

12.5　利用模板美化显示

12.5.1　模板

模板(Template)是一组样板，它将 HTML 元素与 ASP.NET 的控件结合在一起用来定义数据的显示格式，并且由这些格式形成最终的布局。模板相当于框架，在框架中可以放入控件，通过控件与数据绑定，使得这些绑定的数据按照模板规定的格式显示。

模板与样式表(CSS)有联系但也有区别。样式表定义的是某个特定元素的属性(如该元素字体的类型、大小、颜色等)。而模板能够更加深刻地改变控件的整体外观，还能给控件增添新的功能。

通常将模板与样式表结合起来，以便全面改善界面的显示，增强控件的功能。

控件中的模板由头、尾、体三部分组成，分别用头模板(HeaderTemplate)、尾模板(FooterTemplate)和体模板(ItemTemplate)表示。三部分的关系如图 12.13 所示。

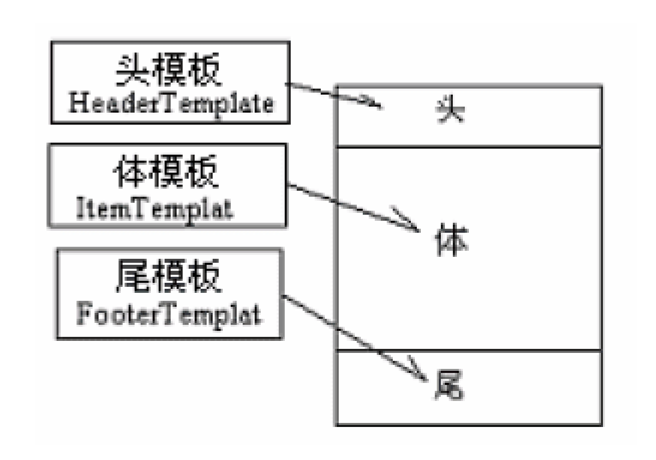


图 12.13　模板的结构

其中，头模板和尾模板用来设置数据标题和尾部显示的内容和格式。这两种模板是选用部分(可采用也可不采用)。体模板是必须选用的，它用来显示数据的主体，当绑定有多条记录时，在体模板中将自动扫描数据源的各条记录，并且按照模板的要求逐条显示出来。体模板有时又可细分为选择模板、编辑模板等，用来定义被选中记录或编辑记录的显

示样式。还可以用交替模板来设置交替记录的不同的样式(例如不同的底色)。

控件通常具有固定的功能和显示界面。如果控件拥有模板功能，就能在不同的情况下自动转换成不同的界面并执行不同的任务，控件的功能从而得到了大大的加强，一个控件可以当多种控件来使用。

下面几章将介绍各种模板的使用方法。下面先介绍在 GridView 控件中对模板的设置方法。

12.5.2　自动套用格式

自动套用格式设置起来最简单，但灵活性不够，功能也不够强。使用方法是：右击窗体页内 GridView 控件，在弹出的快捷菜单中选择【自动套用格式】命令，弹出如图 12.14 所示的对话框。

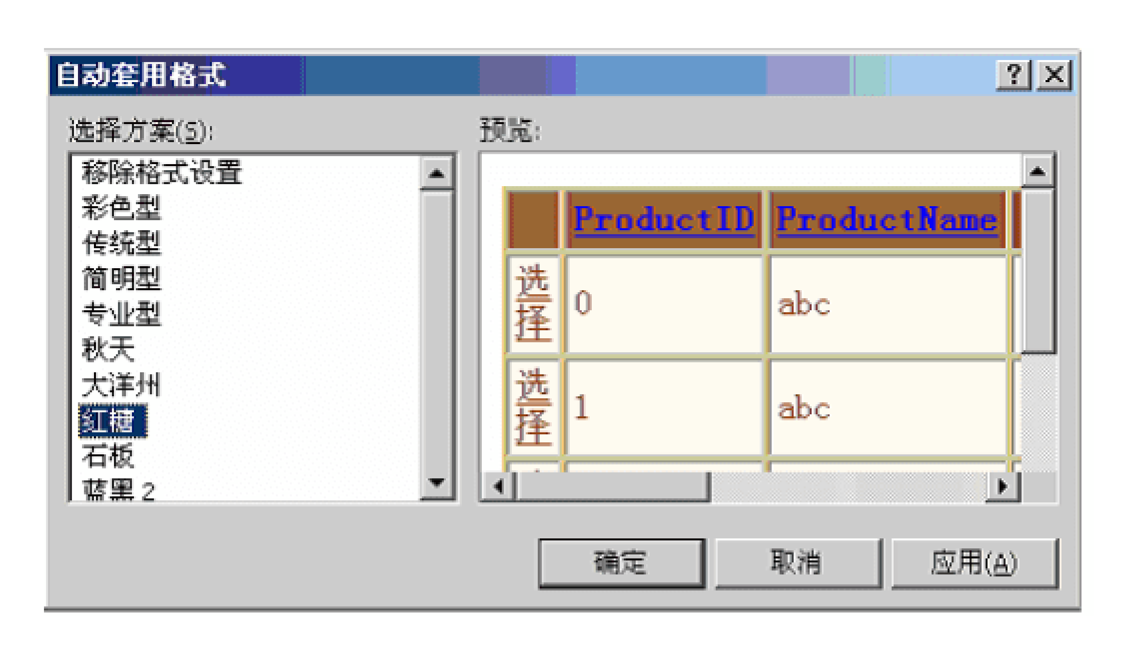


图 12.14　【自动套用格式】对话框

单击左边【选择方案】列表框中的各项，右边的【预览】列表框中将显示出该方案所对应的显示界面。逐个单击左边的方案，直到选择一个合适的方案为止。单击【确定】按钮，即完成了模板的设置工作。

12.5.3　设置模板样式

打开 GridView 控件的【属性】对话框，可以看到 6 个模板样式的选项。

- HeaderStyle：头模板样式。
- ItemStyle：体模板样式。
- FooterStyle：尾模板样式。
- AlternatingItemStyle：交替模板样式。
- SelectedItemStyle：选择项模板样式。
- EditItemStyle：编辑项模板样式。

每种样式都包括底色(BackColor)、边界颜色(BorderColor)、边界样式(BorderStyle)、边界宽度(BorderWidth)、其他样式(CssClass)等，利用这些属性可以定义模板的显示特征。

程序运行时，前 4 种模板的样式即可显示出来。而 SelectedItemStyle 和 EditItemStyle 两种模板只有当用鼠标选中某条记录或者对某条记录进行编辑时，才会显示出来。

12.6　显示记录中的图像

为了显示记录中的图像，先将图像复制到网站的某个目录下，然后在数据表的相关字段中填入这些图像的路径。

假定已经建立了数据表。现在为了在 GridView 控件中显示记录中的图像，需要补充以下步骤。

(1)　将需要显示的图像复制到网站的某个目录下。

假定几个图像复制到网站的情况如图 12.15 所示。

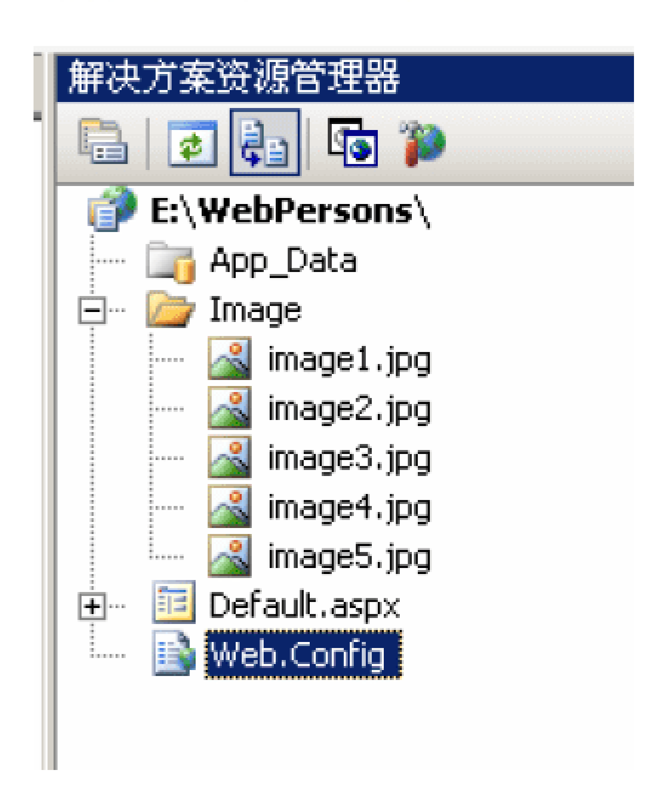
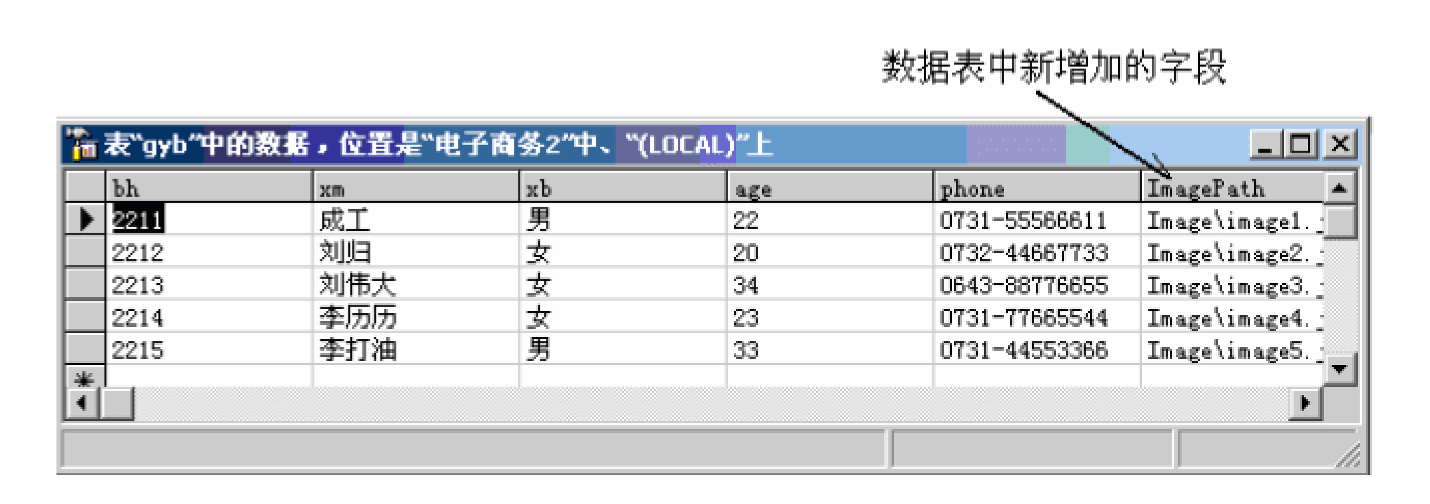


图 12.15　将需用的图像放在网站中

(2)　在数据表中增加【图像路径】字段，并填入各图像在网站中的路径(URL)。数据表的情况如图 12.16 所示。



bh	xm	xb	age	phone	ImagePath
2211	成工	男	22	0731-55566611	Image\image1.
2212	刘归	女	20	0732-44667733	Image\image2.
2213	刘伟大	女	34	0643-88776655	Image\image3.
2214	李历历	女	23	0731-77665544	Image\image4.
2215	李打油	男	33	0731-44553366	Image\image5.

图 12.16　雇员表的部分内容

其中，ImagePath 代表照片(图像)在网站中的路径，应设置成可变长字符串类型。该路径要与图 12.15 中各图像的路径相对应。

(3)　在 GridView 控件的字段编辑对话框中将 ImageField 增加到字段中来，同时删除原有的 ImagePath 字段。

(4)　设置 ImageField 字段的属性。属性的设置情况如图 12.17 所示。

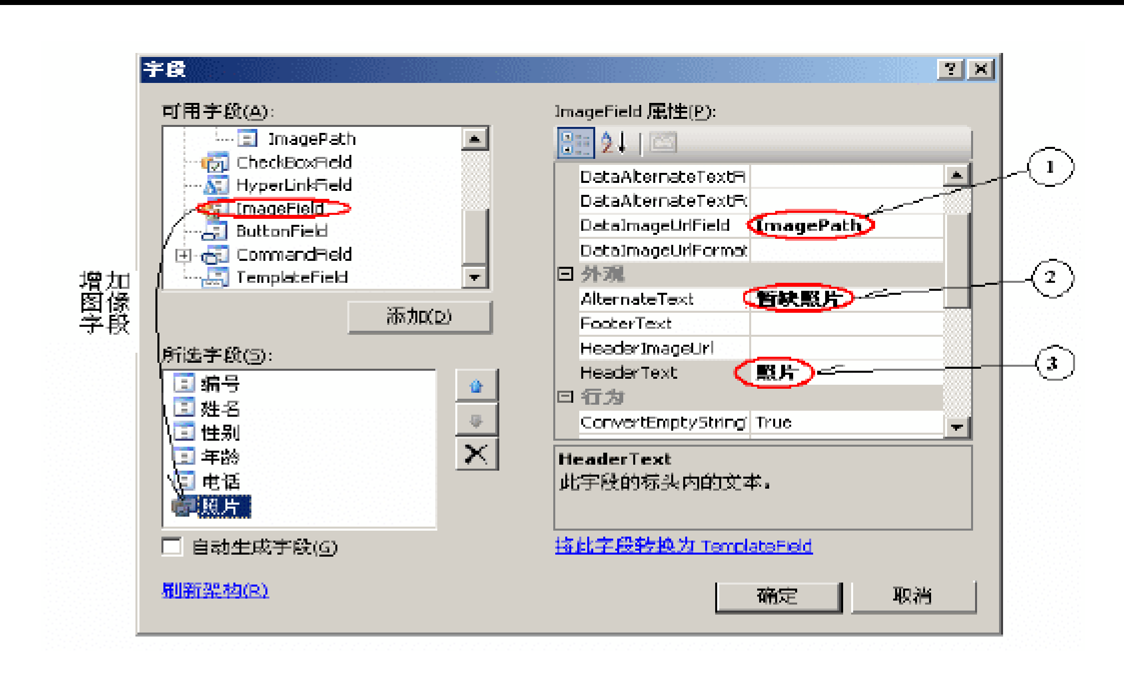


图 12.17　数据表中显示照片的设置

其中，①将 DataImageUrlField 属性与数据表字段 ImagePath 绑定；②在 AlternateText 属性栏写上“暂缺照片”，当图像字段中缺图像时显示此字符串；③将本栏的标题(HeaderText)改成“照片”。

运行时表格的显示如图 12.18 所示。

选择	编号	姓名	性别	年龄	电话	照片
选择	2211	成工	男	22	0731-55566611	
选择	2212	刘日	女	20	0732-44667733	
选择	2213	刘伟大	女	34	0643-88776655	
1 2						

图 12.18　数据表的显示结果

12.7　小　　结

GridView 控件的功能非常强大，不仅能够用于显示数据表中的数据和图像，还能够编辑数据表中的数据(后面介绍)。因此这个控件是本书讲述的重点。

通常情况下，GridView 控件可以通过数据源控件 SqlDataSource 与大型数据库(SQL Sever、Oracle 等)连接，并且选择显示的数据表以及显示的字段等。通过 GridView 的属性还可以直接给数据表分页、排序和执行选择功能。由于在数据源控件中隐含有大量常用的代码，上面这些设置都变得非常简单，几乎不需要增加任何手写的代码。

利用 GridView 模板，还可以进一步美化 GridView 控件的外观，并且增加一些附加功能。

12.8　习　　题

1. 填空题

(1) GridView 控件的基类是 ____________________。

(2) 分页后每页默认的记录是________条。

2. 判断题

(1) GridView 控件只能原样显示数据表中的记录。　(　)

(2) 为了美化显示，GridView 控件的头模板、体模板、尾模板都必须进行设置。　(　)

3. 简答题

(1) 简述 SQL Server 2008 Express Edition 的特性。

(2) 简述 GridView 控件连接并显示数据表的过程。

(3) 什么是模板?

(4) 为了显示数据表记录中的照片，应做哪些准备工作?

(5) 为了显示记录中的照片，应该如何设置 GridView 控件的属性?

4. 操作题

(1) 将 GridView 控件连接并显示 NorthWind 样板库中的 Prducts 数据表。要求:

- 分页，每页 5 条记录。
- 排序。
- 被选中的记录底图呈浅蓝色。
- 记录隔行改变底色。

(2) 自己定义一张雇员表，包括编号(bh)、姓名(xm)、性别(xb)、年龄(nl)、照片(zp)等字段并且输入若干条记录。通过 GridView 控件连接和显示该数据表，要求除与(1)题相同外，还要显示照片。

第 13 章　数据库查询与同步

数据库查询就是在数据库中查找符合条件的记录。例如，在书库中查找某图书的情况；在人事档案中查看某个人的相关数据；在仓库中查看某产品的存储量等。数据库的其他一些操作，例如修改、删除记录等也常常是在查询的基础上进行的。因此查询是最重要也是使用得最为频繁的数据库操作。

数据库多表同步也是应用程序中使用得比较频繁的操作。由于对数据表同步功能的设计与查询的设计有一些共同点，因此放在同一章中来讲述。

ASP.NET 3.5 在原来版本的基础上对查询和同步的设计都进行了很大的改进，不仅对原有控件作了进一步封装，还增添了新的控件，从而使得设计的过程变得非常简单，功能变得更加强大。

本章讲述上述两个问题时，采取的步骤为先讲述一般原理，然后结合示例讲明应用步骤，最后进行代码分析。具体内容包括：

- 数据库查询：数据库查询语句、单一条件查询、选择条件查询、多条件的组合查询。
- 数据表同步：在同一窗体页中父/子表的同步、在不同窗体页中父/子表的同步。

13.1　数据库查询

13.1.1　数据库查询语句

数据库查询通常都是利用 SQL 语句进行的。在 SQL 查询语句中包括一些待定参数，这些待定参数就是查询条件。待程序运行时，给出不同的参数就可以获得不同的查询结果。

SQL 的查询语句的格式如下。

```
Select * From 数据表名
Where ( 字段 1 < @待定参数 1 AND 字段 2 < @待定参数 2… )
```

其中 Select、From、Where 都是保留字。Where 后面为查询条件。

给待定参数赋值的方法在 ASP.NET 类库中进行了多次重载。其中最简单的格式是

```
命令组件名.Parameters.Add("@待定参数",数据源);
```

其中数据源可以是某个常数或者某个控件的值。当数据表中字段类型属于字符串或者整型数时，利用上述语句即可。但是数据表中字段的类型很多，为了适应各种类型数据的需要，系统还提供了通用的赋值语句。格式如下。

```
组件名.Parameters.Add(new SqlParameter("待定参数名",SqlDbType.类型));
组件名.Parameters["待定参数名"].Value = 实际参数;
```

其中第一句确定了参数的类型，第二句给参数具体赋值。例如，访问样板库 Northwind 中的 Products 表，将其中的字段 ProductID 作为查询条件，并将待定参数设成@ProdID，给待定参数赋值的语句如下。

```
sqlCommand1.Parameters.Add(new SqlParameter("@ProdID",SqlDbType.int));
sqlCommand1.Parameters["@ProdID "].Value = TextBox1.Text;
```

在 ASP.NET 1.x 中虽然提供了智能提示，但实际设计时要正确确定参数的类型仍然不怎么容易。现在 ASP.NET 3.5 新版本作了进一步简化。它将参数类型的确定、SQL 查询语句的生成以及执行查询语句等项工作都隐含在数据源控件的内部。设计时只需按照系统的提示进行选择即可，不再需要编写 SQL 语句以及给参数赋值的语句。

13.1.2　单一条件查询

所谓简单查询是指单一条件的查询。下面将首先用一个 TextBox 控件提供查询条件，然后改用下拉列表框(DropDownList)提供查询条件。

1. 数据源控件的设置

在 ASP.NET 3.5 中进行数据库查询的关键在于设置数据源控件(DataSource Control)。

下面以对 Northwind 样板库中的 Products 表的查询为例，说明给数据源控件设置的方法。要求根据类型字段(CategoryID)的值查出表中该类型的产品。

首先在窗体页中放入 TextBox 与 GridView 控件。其中 TextBox 控件用于输入 CategoryID 的值；GridView 控件用于显示查询结果。

为了建立与数据库的连接，需要配置数据源控件。在配置数据源控件的过程中当打开选择字段的对话框时，出现的界面如图 13.1 所示。

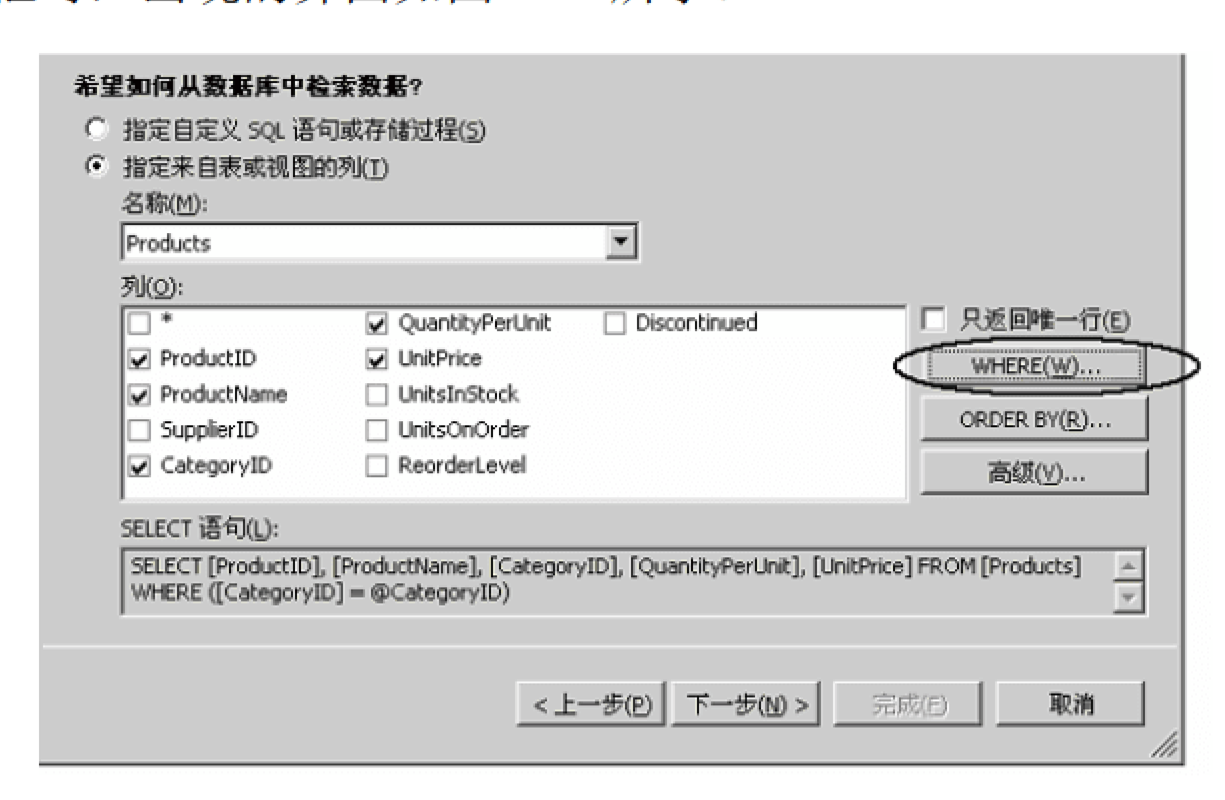


图 13.1　单击 WHERE 按钮打开查询设置语句

单击 WHERE 按钮，将弹出如图 13.2 所示的对话框。

图 13.2 设置查询参数

在【列】下拉列表框中选择待定参数对应的字段；在【运算符】下拉列表框中确定条件的运算符(大于、小于、等于)；在【源】下拉列表框中选择参数的来源。参数的来源包括以下几种。

- Control：从控件中获得参数。
- Cookie：从 Cookie 对象中获得参数。
- Form：从窗体页中获取参数。
- Profile：从客户配置文件中获取参数。
- QueryString：从上一个网页用 get 提交时传来的语句中获得参数。
- Session：从 Session 对象中获得参数。

选择 Control 选项，然后在【控件 ID】下拉列表框中选择 TextBox1。单击【添加】按钮，在【SQL 表达式】以及【WHERE 子句】文本框中将出现相应的字符串，如图 13.2 所示。这段字符串表明待定参数为

```
[CategoryID] = @CategoryID2
```

待定参数值来源于 TextBox1.Text。

返回原有窗口，可以看到一个 SQL 的查询语句已经在【SELECT 语句】文本框中形成，如图 13.3 所示。

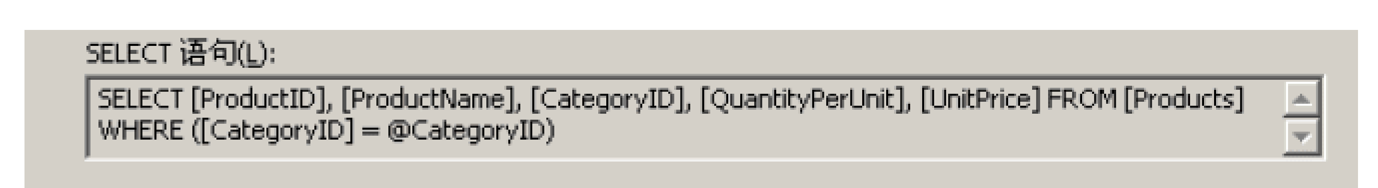


图 13.3 设置后产生的 SQL 语句

单击【下一步】按钮，在新打开的对话框中单击【测试查询】按钮，并输入待定参数以查看查询的结果，如图 13.4 所示。

如果查询的结果符合设计要求，说明对查询的设计已经完成。

2. 用 DropDownList 提供查询条件

现在用 DropDownList(下拉列表框)控件代替原来的 TextBox 控件。新控件可以和另一个数据表进行数据绑定，从该表中获取查询条件。

现在将 DropDownList 控件与类型表(Categories)进行数据绑定，其方法与前面

GridView 基本相同。即单击 DropDownList 右上角的◀图标，选择【<新建数据源>】命令，然后建立连接(可以利用上面已经创建的连接对象)，选择数据表及字段(这里只需要选择 CategoryID、CategoryName 两项)等。当选择完字段以后，将弹出如图 13.5 所示的对话框。

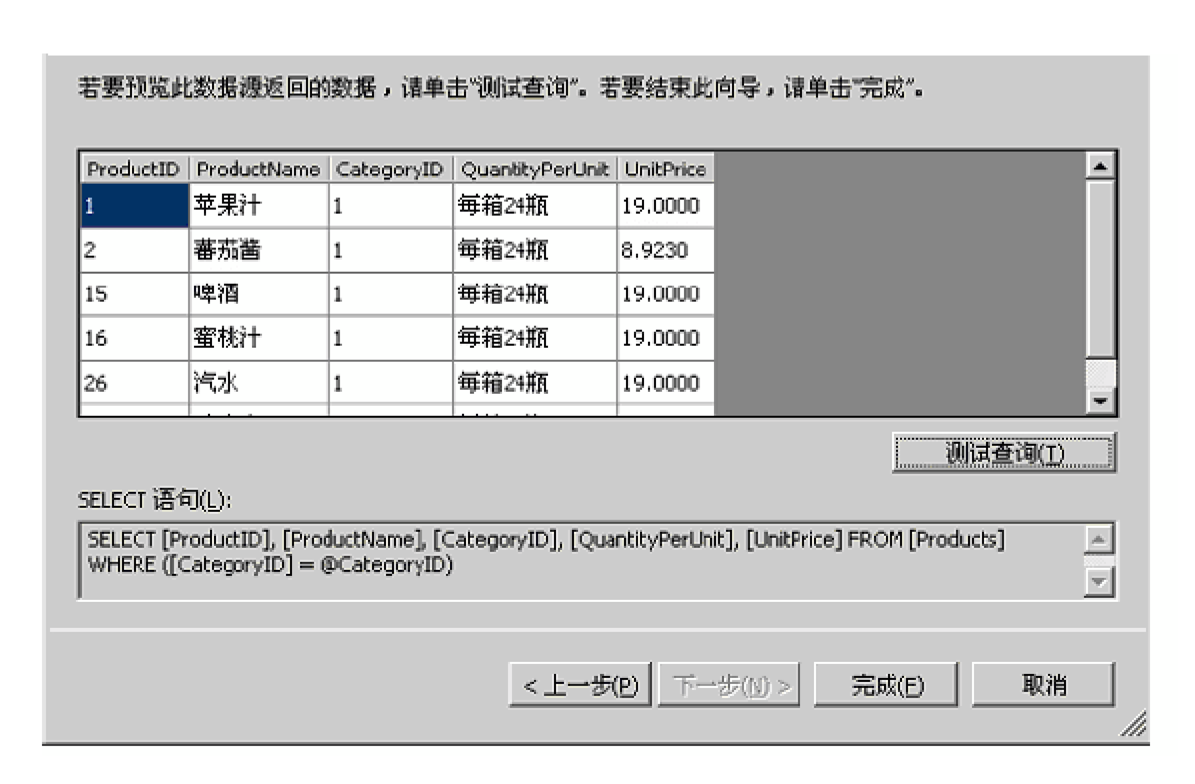


图 13.4　查询 CategoryID=1 的结果

选择其中的 CategoryName 属性用于对外显示，属性 CategoryID 用于内部控制。

利用 DropDownList 提供查询条件时的结果如图 13.6 所示。

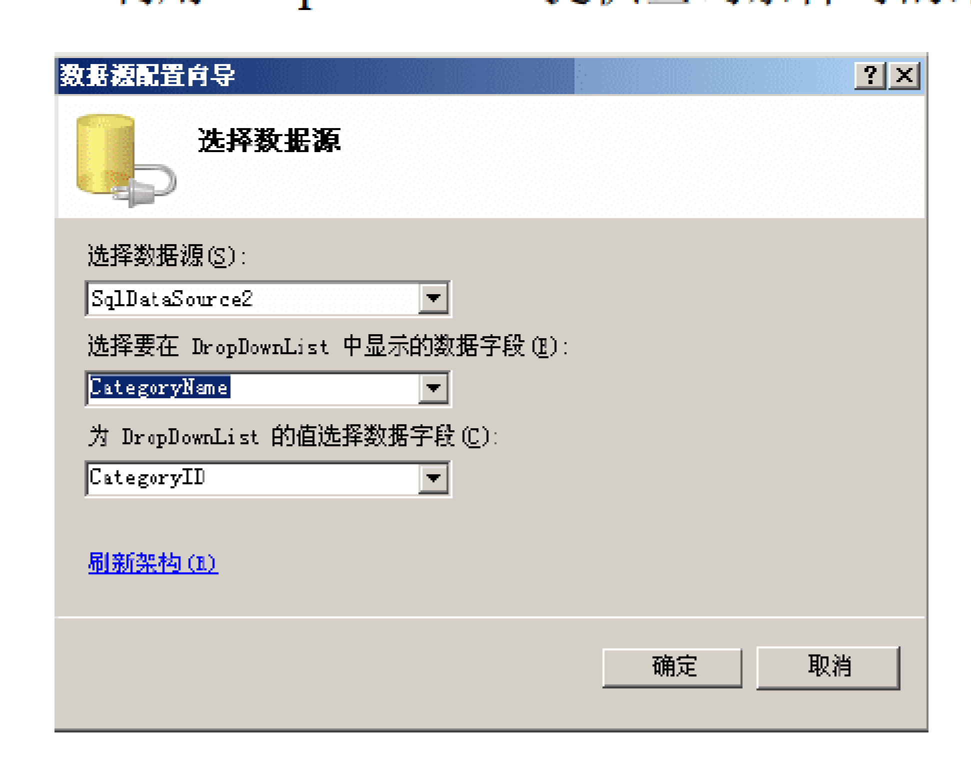


图 13.5　为下拉列表框设置属性



编辑	产品ID	产品名	类型ID	单元数量	单价
选择	20	德国奶酪	4	每箱24瓶	19.0000
选择	21	民众奶酪	4	每箱24瓶	19.0000
选择	27	温性奶酪	4	每箱24瓶	29.8000

图 13.6　查询结果

13.1.3　选择条件查询

在窗体页中设置一套控件，但允许选择多种不同的条件进行查询，这样的设置能给客户的操作带来一些方便。比如在一些出版社的网页中，允许读者选择【书名】来查询图书，也允许选择【作者名】来查询图书，就常采用这种结构方式。

为了实现选择条件查询，网页中可以设置如下控件。

- 一个下拉菜单控件(DropDownList)：用来提供选择条件。
- 一个输入控件(如 TextBox)：用来输入某条件的值。
- 一个 GridView 控件：用来显示查询结果。
- 一个按钮(Button)：用来编写事件代码。

- 两个或两个以上数据源控件(SqlDataSource)：分别用来生成不同条件的查询语句。

例如查询一个如第 12 章中创建的数据表。在下拉菜单中提供【按照年龄查询】和【按照性别查询】两种选择。下拉菜单的设置如图 13.7 所示。

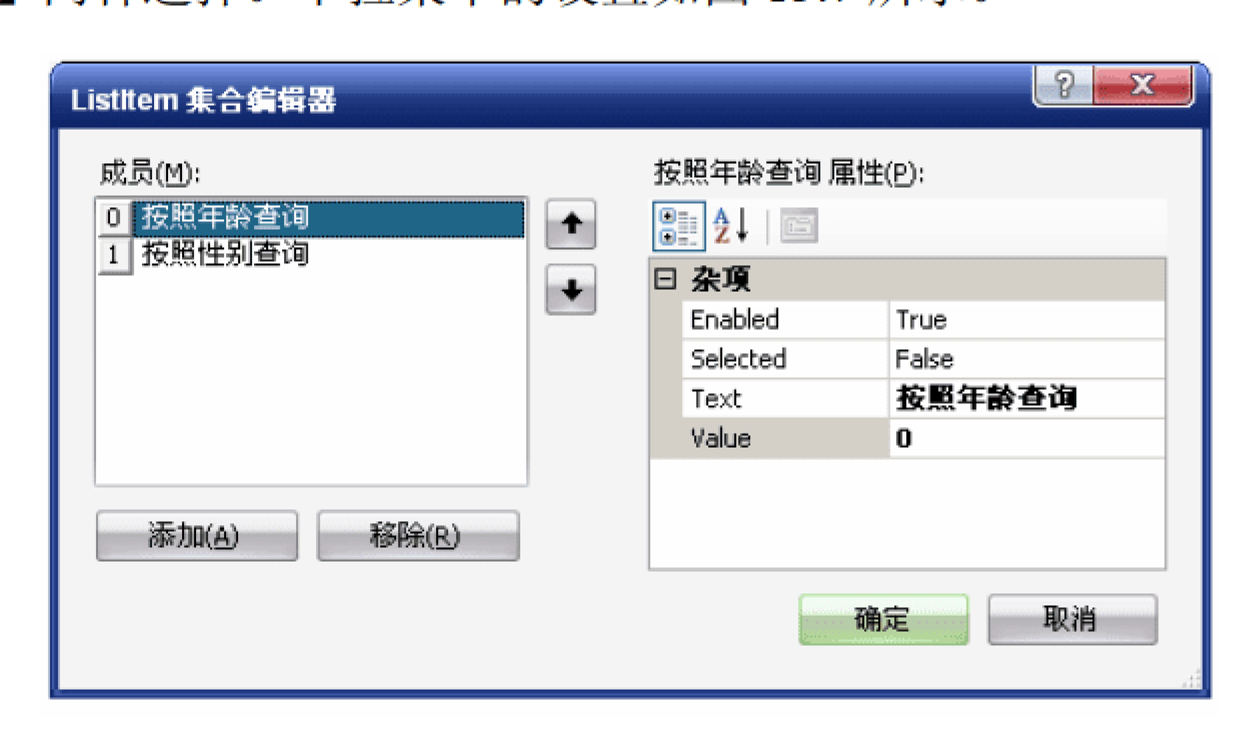


图 13.7　在下拉菜单中提供选择条件

将一个数据源(SqlDataSource1)以年龄小于某个值作为查询条件。经过配置后的查询语句如下。

```
 SelectCommand="SELECT [bh], [xm], [xb], [age], [phone] FROM [gyb]
WHERE ([age] < @age)"
```

将另一个数据源(SqlDataSource2)以性别作为查询条件，经过配置后的语句如下。

```
SelectCommand="SELECT [bh], [xm], [xb], [age], [phone] FROM [gyb]
WHERE ([xb] = @xb)"
```

为按钮的 Click 事件编写如下代码。

```
protected void Button1_Click(object sender, EventArgs e)
      {
        if (DropDownList1.SelectedValue == "0")
                //按年龄进行查询时，引用第一个数据源控件
            GridView1.DataSourceID = "SqlDataSource1";
        else
            GridView1.DataSourceID = "SqlDataSource2";
                //按性别进行查询时，引用第二个数据源控件
  }
```

13.1.4　多条件的组合查询

有时需要将多个条件组合起来进行查询。为此需要多设几个输入控件(例如，设置多个 TextBox)以提供查询条件，并在配置数据源控件时将设置待定参数的操作重复多次(具体方法见图 13.2)，就自动形成了多个条件之间的“与”关系。

多条件之间的“与”关系要求只有在输入了全部条件时，才能进行查询。如果某个(或某些)条件没有输入时，“空”仍然将作为条件与其他条件相与，结果使得查询失败。这种情况给客户造成了一定的困难。因为在很多情况下，客户给不出所有的条件。

应该允许在缺少某些条件的情况下仍然能够正常进行查询，只是查询的结果范围可能

更宽一些而已。为了实现这一功能，需要做以下两方面的改进：

(1) 在设定查询参数时必须给每个输入的控件设定“默认值”。默认值在图 13.2 中设置，它代表控件没有输入时的默认值。为了避免混淆，这个值应该远离可能出现的查询参数。

(2) 在网页的【源】窗口中用手动方法修改数据源的 SQL 查询语句。

例如，修改前的查询语句是

```
Select * From 数据表名
Where ( 字段 1 = @待定参数 1 AND 字段 2 = @待定参数 2… )
```

修改后的语句为

```
Select * From 数据表名
Where (( 字段 1 = @待定参数 1 ) OR (@待定参数 1= 默认值 1 )) AND
      ((字段 2  =  @待定参数 2) OR (@待定参数 2 = 默认值 2)) AND … )
```

由于 AND 的优先级高于 OR，因此应该用括号将 OR 的操作括起来。执行上述语句时，一旦某个控件的输入出现空缺，“@待定参数= 默认值”的结果为 true，因此这个控件的空缺不会影响其他条件的正常查询。

现在用一个示例说明上述过程。

假定仍使用第 12 章创建的数据表。用年龄和性别相与(AND)作为条件进行组合查询。网页中的界面如图 13.8 所示。



bh	xm	xb	age	phone
2211	成工	男	22	0731-55566611
2215	李打油	男	33	0731-44553366

图 13.8　缺少年龄条件下的组合查询

修改前的查询语句是

```
SelectCommand="SELECT [bh], [xm], [xb], [age], [phone] FROM [gyb]
WHERE (([age] < @age) AND ([xb] = @xb))"
```

为了允许在缺项的情况下仍能正常查询，按照以下步骤进行。

(1) 重新配置数据源控件(SqlDataSource)，为每个输入控件设置默认值。在这个示例中，给 TextBox1(用于输入年龄的控件)的默认值为-1；给 TextBox2(用于输入性别的控件)的默认值为“*”。

(2) 用手动方法修改 SQL 查询语句。修改后的语句如下。

```
SelectCommand="SELECT [bh], [xm], [xb], [age], [phone] FROM [gyb]
WHERE ((( [age] < @age ) or ( @age=-1 )) AND (([xb] = @xb) or ( @xb='*' )))"
```

如果程序运行时，只输入了性别“男”，没有输入年龄，结果将把所有的男性都显示出来。

如果想将查询条件改用其他形式，例如想用“或(or)”关系代替“与(and)”关系，应该稍微改变一下前面的设置方法，以调用系统提供的其他功能。

下面举例说明改变的具体步骤。在图 12.10 所示对话框的左上角选择【指定自定义 SQL 语句或存储过程】选项，单击【下一步】按钮，在弹出的对话框中单击【查询生成器】按钮，将弹出如图 13.9 所示的对话框。

在查询生成器中，单击【添加】按钮，出现【添加表】对话框。在【添加表】对话框中选择数据表名(图中选择了 Products)，单击【添加】按钮，然后在【查询生成器】对话框中选择字段并且利用表格的其他字段构成相应的 SQL 语句，如图 13.10 所示。

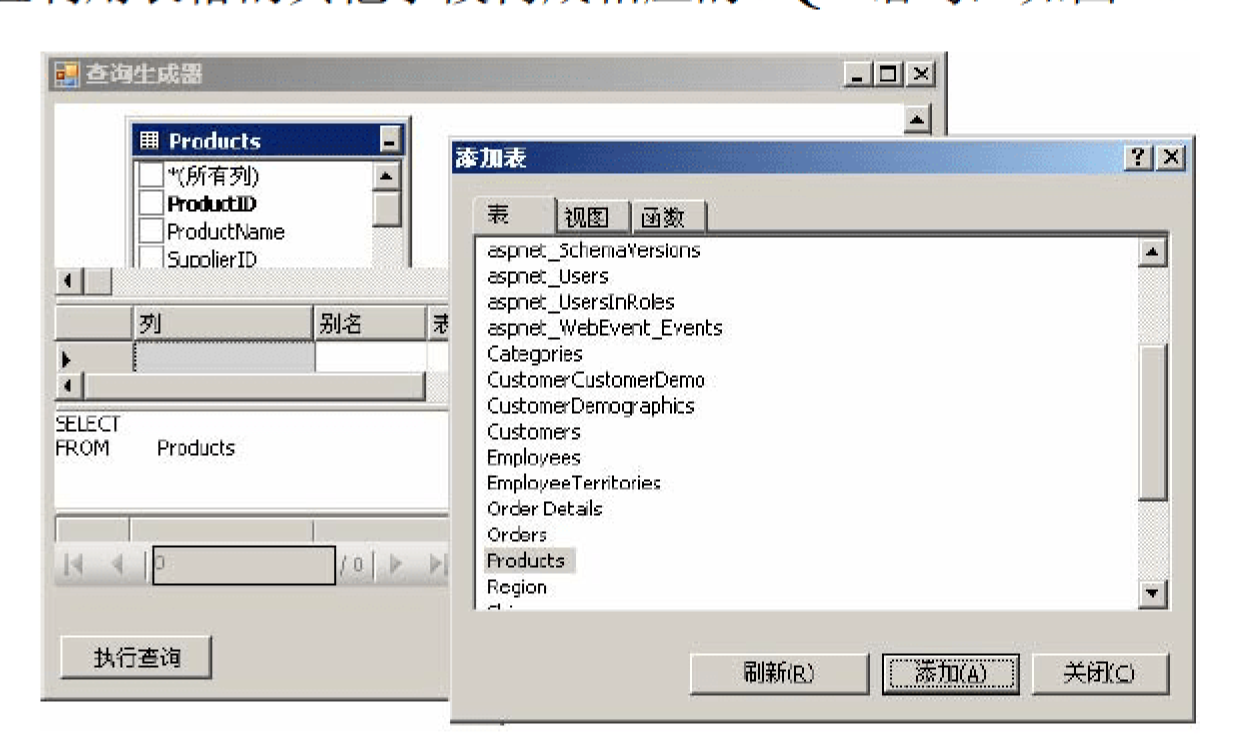


图 13.9　打开查询生成器

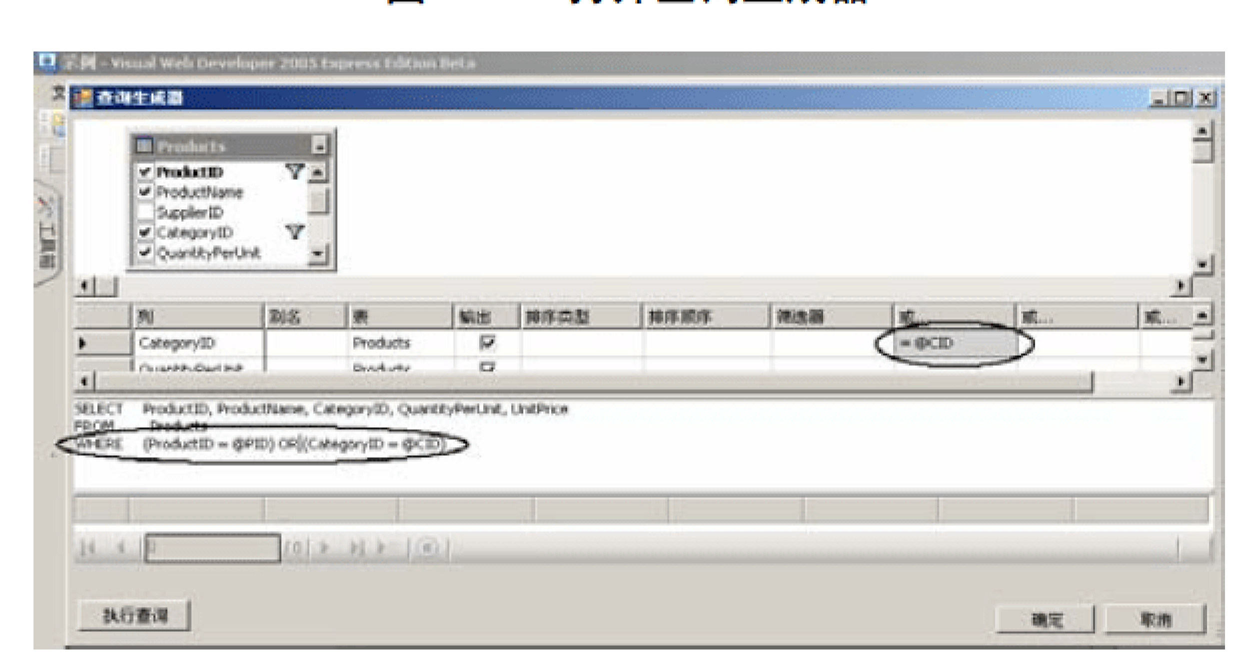


图 13.10　设置查询条件的“或”关系

注意：图中将用于查询的参数放在“或”字段的下面，就构成了 SQL 语句中用“OR”将参数组成的查询条件。

13.2　数据表同步

13.2.1　概述

数据之间的简单关系也许用一张表来表示就足够了，但是如果数据之间存在着比较复杂的关系时，仍然采用一张表来处理，就可能带来大量的数据冗余。根据数据库的“关系规范”理论，有的情况下，将一个复杂的关系分解为一组子关系时，可以减少冗余，优化

结构。

比如，“一对多”的关系就是一种比较复杂的关系。一个客户有多张订单，每张订单上包括多项货物的订购；或者在学校中，一个教学班有几十名学生，每名同学有多门功课的成绩等。上述关系最好用多张表来表示，在这里“一”方通常被称为“父表”，“多”方通常被称为“子表”，父表与子表之间通过同步字段实现同步。

父、子表之间虽然在逻辑上存在一定的联系，但是它们都是独立的数据表，对每张表的访问也是一个独立的过程。在这种情况下，如何实现它们之间的同步？下面分别介绍两种实现同步的方法：在同一窗体页中实现父、子表的同步；在不同窗体页中实现父、子表的同步。

13.2.2 同一窗体页中父、子表同步

1. 实现步骤

为了在同一个窗体中实现父、子表的同步，需要在窗体中设置两个 GridView、两个数据源控件。每个 GridView 通过自己的数据源控件连接到父表或子表。对于父表来说，在 GridView 中只需要增加一个【选择】按钮，并确定被选择行的样式即可；对于子表来说，主要是根据父表选择的字段进行查询，并显示查询的结果。

下面以 Northwind 样板数据库中的 Categories 表作为父表，Products 表作为子表，说明利用两个 GridView 实现同步的过程。

先将两个 GridView 控件放入窗体中，分别通过自己的数据源控件连接到父表和子表，并按图 13.11 所示的方式进行设置。

重新配置 GridView(子表)的数据源控件，设置 SQL 的查询语句，查询条件来自父表的字段(这里使用 CategoryID)。设置的方法如前所述。只不过将【控件 ID】中的 TextBox1 改为 GridView1。

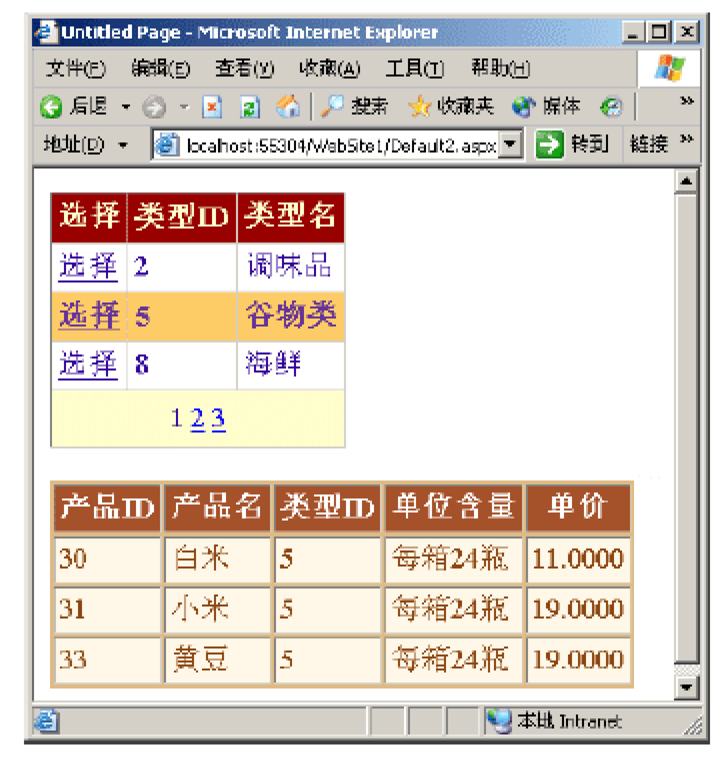


图 13.11 同一窗体中两个 GridView 同步

2. 代码分析

用于主表的数据源控件的代码与一般代码没有什么区别，关键在于子表查询时如何确定待定参数。子表数据源控件的代码如下。

```
<asp:SqlDataSource ID="SqlDataSource2"
Runat="server"
    SelectCommand="SELECT [ProductID], [ProductName], [CategoryID],
     [QuantityPerUnit],[UnitPrice] FROM [Products]
    WHERE ([CategoryID] = @CategoryID)"
    ConnectionString="<%$ ConnectionStrings:AppConnectionString1 %>">
    <SelectParameters>
        <asp:ControlParameter Name="CategoryID" DefaultValue="1"
        Type="Int32 ControlID="GridView1"PropertyName="SelectedValue">
        </asp:ControlParameter>
    </SelectParameters>
 </asp:SqlDataSource>
```

在子表的数据源控件中，用 SelectCommand 查询语句“WHERE ([CategoryID] = @CategoryID)”指出了待定参数，同时在下面的参数语句中确定了这个待定参数的来源。ControlID="GridView1"表明该参数来源于 GridView1，而 GridView1 就是显示主表的控件。

13.2.3 不同窗体页中父、子表同步

1. 实现步骤

在不同网页中实现父、子表同步的原理与前面的方法相同，都是在子表中将父表传来的字段作为参数进行查询，然后显示查询结果。和前面所讲方法的最大区别在于子表获得参数的方式不同。因为要从不同的网页中获取数据，子表要通过 QueryString 属性从 URL 中提取。

同步设置的方法分为对父表的设置和对子表的设置两方面：从父表方面来看，要求调用子表的同时将同步条件(选择的字段)附在调用的 URL 字段后面；从子表方面来看，应利用数据源控件的 QueryString 属性来获取条件以便进行查询，从而达到父、子表同步的目的。

下面通过示例来说明上述同步过程。

假定将 SQL Server 的样板库 Northwind 中的 Categories(类型表)作为父表，Products(产品表)作为子表，两表之间通过 CategoryID(类型 ID)字段取得同步。同步的情况如图 13.12 所示。

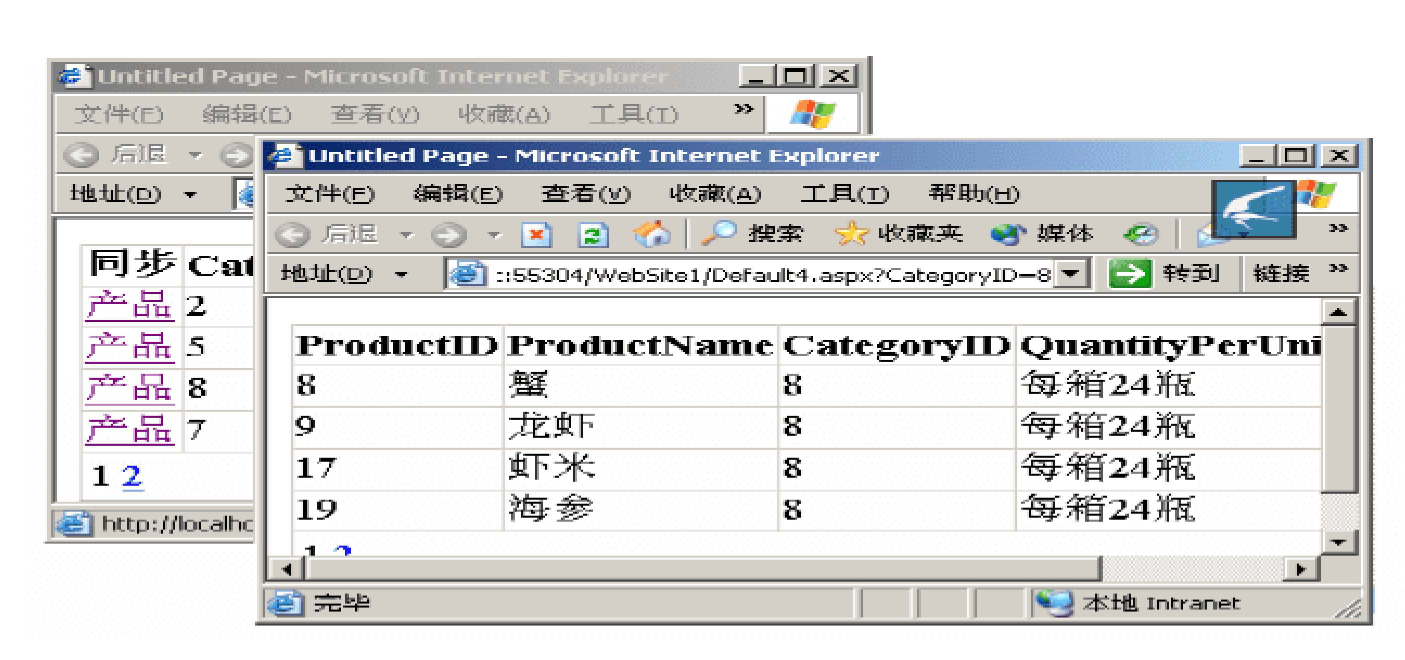


图 13.12 不同网页之间的同步

2. 为父表设置属性

为父表的 GridView 的字段(Columns)属性设置的方法如图 13.13 所示。

① 将 HyperLinkField 字段增加到窗体中来。

② 给增加的字段标题及字段命名。

③ 确定子表网页的位置。

④ 确定子表窗口放置的位置。

⑤ 确定同步的字段。

⑥ 确定调用子表网页的格式。这里使用的格式是

```
~/Default2.aspx?CategoryID={0}
```

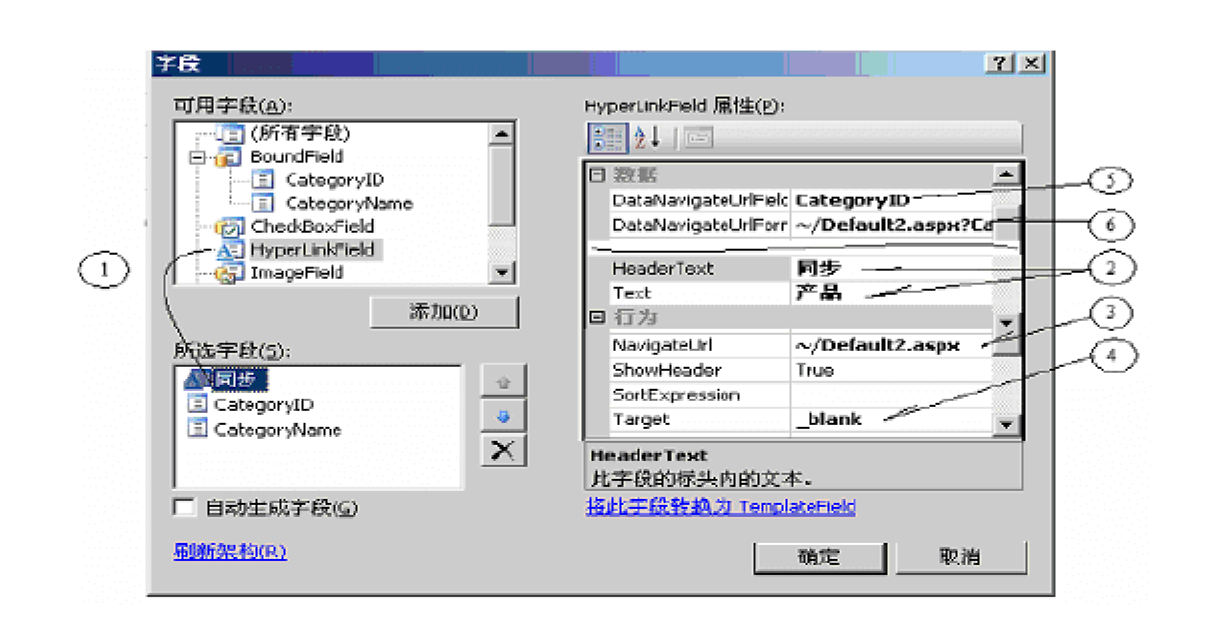


图 13.13　父窗体的设置

3. 重新配置子表的数据源控件

在子表的 GridView 中重新配置数据源控件，选择 Where 后的设置如图 13.14 所示。其中最大的不同是在【源】下拉列表框中选择 QueryString 项。这表明，子表将通过 Request.QueryString 获得从父表传来的参数。

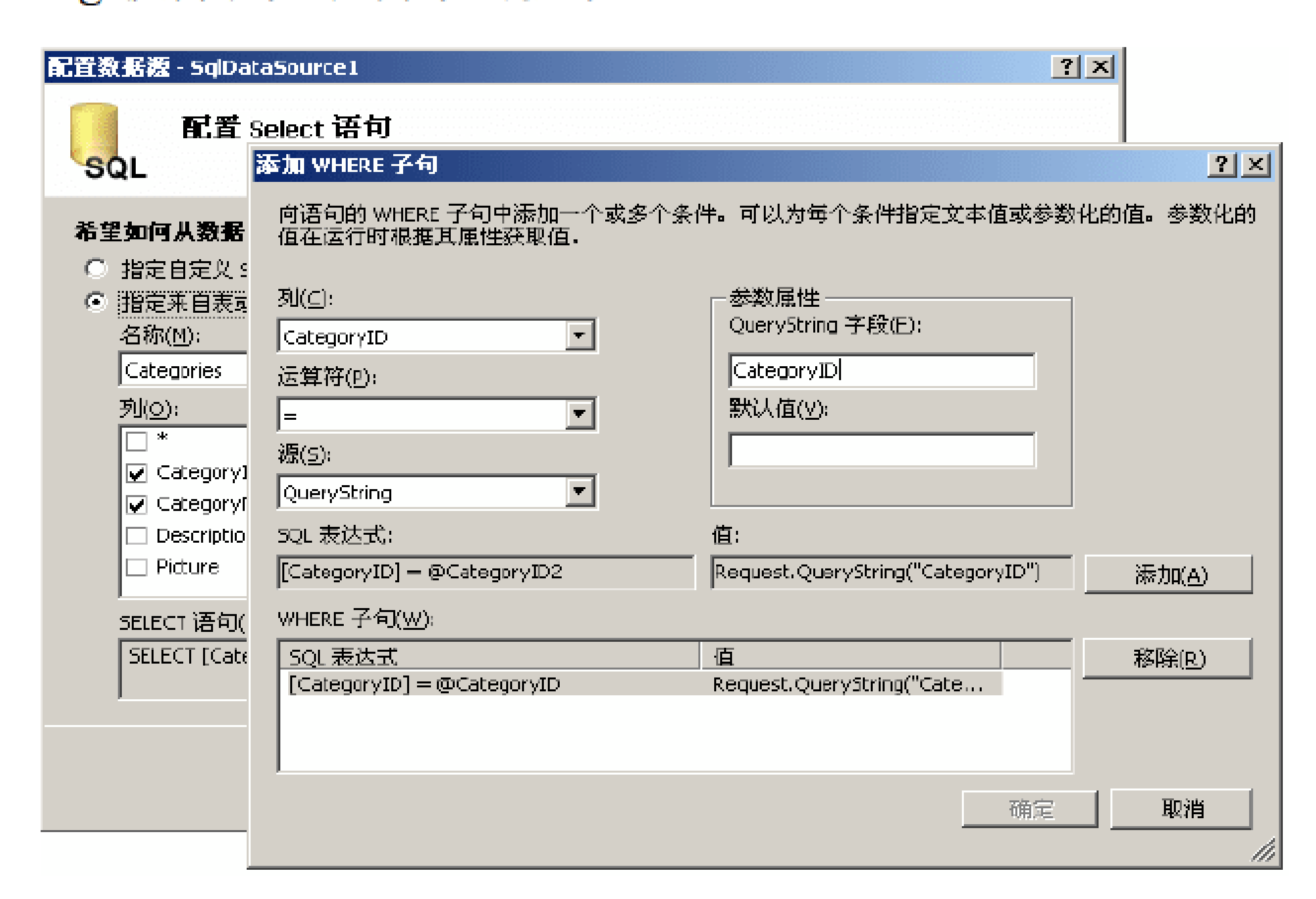


图 13.14　子窗体的设置

13.3　合并多表显示

有时将“一对多”的关系分解成多表，并取得同步可以优化结构，减少冗余，需要时又可以将它们合并到一张表中显示。下面讲述将多表合并成一张表中显示的方法。

在 Northwind.mdf 样板库中，类型表(Categories)与产品表(Products)之间就存在着一对多的关系(一种类型对应于多种产品)，它们之间的同步字段是 CategoryID。合并显示的步骤如下。

(1)　在网页中设置控件并建立与数据库的连接。

(2)　当出现图 12.10 所示的界面时选择【指定自定义 SQL 语句或存储过程】项，单击【下一步】按钮，在弹出的对话框中单击【查询生成器】按钮，弹出如图 13.15 所示的对话框。

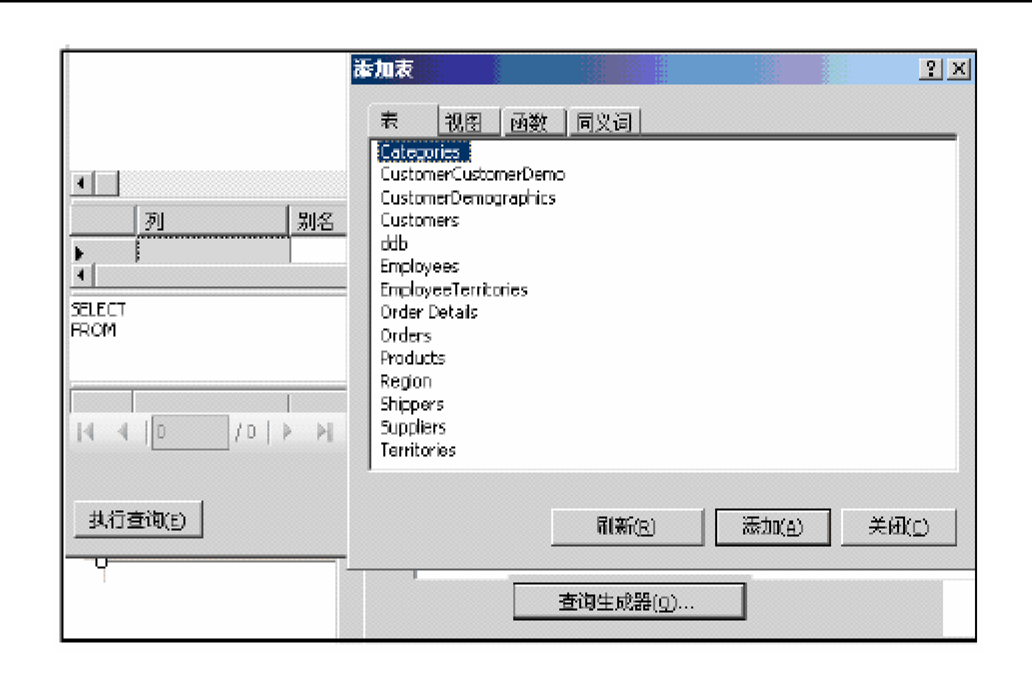


图 13.15　选择数据表

(3) 选择 Categories 与 Products 两张表，并选择需要显示的字段以及排序的要求等，如图 13.16 所示。

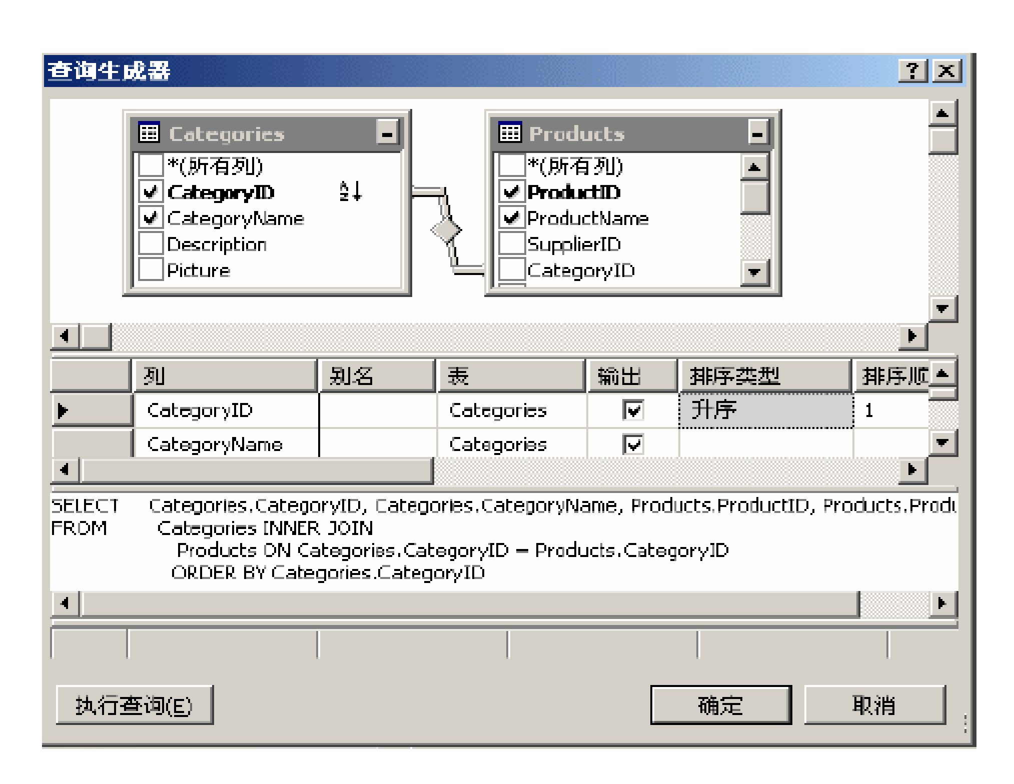


图 13.16　选择多表

注意：在图 13.16 所示对话框的【别名】栏中填写显示的字段名。

对话框的下方显示出多表合并显示的 SQL 语句。它们是

```
  SELECT Categories.CategoryID, Categories.CategoryName, Products.ProductName,
Products.QuantityPerUnit, Products.UnitPrice
FROM Categories INNER JOIN
  Products ON Categories.CategoryID = Products.CategoryID
ORDER BY Categories.Category ID
```

(4) 对字段略加编辑并运行程序时显示的界面如图 13.17 所示。

类型ID	类型名	产品ID	产品名
1	饮料	1	苹果汁
1	饮料	2	蕃茄酱
1	饮料	24	健力宝饮料
1	饮料	34	橘子汁
1	饮料	35	红碗汁
1	饮料	38	中国咖啡
1	饮料	39	啤酒

图 13.17　多表合并显示

13.4 小　　结

数据库查询和同步都是使用得非常频繁的操作，因此是应用程序中不可缺少的重要组成部分。两种方法都需要用到 SQL 的 WHERE 查询语句，因此确定 WHERE 的条件并获取 WHERE 条件的值是问题的关键。

在同一窗体中两个表格的同步与查询操作非常相似。在不同窗体之间的同步却有着较大的区别，这是因为在这种情况下，父窗体在打开子窗体的同时还要将同步的参数传出去，而子窗体则需要利用 Request.QueryString 方法从传来的 URL 中获取参数，并且根据这些参数进行查询以达到同步的目的。

将一张数据表分解成若干表，然后进行同步，这种配置有时可以减少冗余，便于网页的保护和操作。需要时还可以将这些多表合并在一张表中显示。

13.5 习　　题

1. 填空题

(1) 给待定参数赋值的语句中通用的格式是

```
组件名.Parameters.Add(new SqlParameter("________",SqlDbType.____));
组件名.Parameters["待定参数名"].Value =_______ ;
```

(2) 同一网页中两表同步时，作为父表的 GridView 控件中只需增加一个______按钮，并为被选择的行设置样式。作为子表的 GridView 只需将___________控件作为提供查询条件的控件即可。

(3) 在不同网页中进行同步时，作为子表的网页应该利用____________________方法获取从父表传来的同步参数。

2. 选择题

(1) 当进行父/子表之间的同步时，父表与子表记录之间常常是一种__________的关系。

A. 多对多　　B. 一对多　　C. 一对一　　D. 多对一

(2) 下面是在文件系统网站中父表网页发出的同步指令。其中带下划线的部分代表_______________。

http://localhost:3018/Website1/<u>Default2.aspx?Category=2</u>

A. 返回的数据　　B. 打开的新网页

C. 传来的参数　　D. B+C

3. 判断题

(1) 当两张表需要同步时，两张表中都必须有同步字段。同步字段的名字可以不同，但类型必须相同。　　(　　)

(2) 利用下拉列表框提供条件进行查询时，下拉列表框与 GridView 控件可以共用一

个数据源控件。 (　　)

(3) 在同一个网页进行同步时，两个 GridView 控件可以共用一个数据源控件。

(　　)

4. 简答题

(1) 为什么说数据查询是使用得最多的操作?

(2) 简述在对不同网页的数据表实现同步时，作为父表应做哪些设置。

(3) 简述在对不同网页的数据表实现同步时，子表如何获取父表传来的参数。

5. 操作题

(1) 设计一个简单的查询网页。要求从下拉列表框控件中提供查询条件，用 GridView 控件显示查询结果。

(2) 设计一个多条件选择查询的网页。

(3) 设计一个组合查询网页，允许在输入缺项的情况下，仍能够正常查询。

(4) 设计一个组合查询的网页，要求设置多个查询条件，条件之间包括“与”和“或”关系。

(5) 将 Northwind 样板库中的 Categories 作为父表，Products 作为子表，在同一网页中实现父/子表之间的同步。

(6) 将 Northwind 样板库中的 Categories 作为父表，Products 作为子表，在不同网页之间实现父/子表之间的同步。

第 14 章　编辑数据表

随着情况的发展和变化，数据表中的某些数据必然需要做相应的改变。改变数据的操作不外乎增、删、改数据三种(“改”有时又称为“更新”)。将这三种操作归结为“编辑”工作。如何设计对数据表的编辑程序，是应用程序设计中不可缺少的部分。

本章将讲解对数据表编辑的设计方法。在本章中将要讲解的内容包括以下几个方面:

- 数据表编辑的 SQL 语句。
- 使用 GridView 控件更新数据表。
- 使用 GridView 控件的列模板。
- 使用 GridView 控件增添记录。
- 使用 DetailsView 控件。

14.1　数据表编辑的 SQL 语句

数据表的编辑工作都是利用 SQL 语句进行的。SQL 语句最基本的格式如下。

(1)　增添记录的语句：

```
INSERT INTO 表名
(字段名 1, 字段名 2…) VALUES ( 待定参数 1, 待定参数 2…)
```

插入新记录时，新记录中各字段的值由各个相应的待定参数指定。在 SQL Server 数据库中，待定参数通常都采用“@参数名”的形式。

(2)　删除记录的语句：

```
DELETE 表名
Where ( 关键字= 待定参数 )
```

语句表明需要删除一条关键字的字段等于待定参数的记录。

(3)　修改(更新)记录的语句：

```
UPDATE 表名
SET 字段名 1 = 待定参数 1, 字段名 2 = 待定参数 2, …
Where ( 关键字 = 待定参数 )
```

语句表明需要修改一条关键字等于待定参数的记录，各字段修改后的值由相应的待定参数(待定参数 1、待定参数 2 等)指定。

(4)　执行 SQL 命令的方法：

在 ASP.NET 的应用程序中，通过命令组件调用相应的方法来执行 SQL 命令。这些方法如下。

- ExecuteNonQuery()方法：用于执行更新、增添或删除数据表中的记录，执行后不返回执行结果。
- ExecuteReader()方法：用于执行查询操作，并将查询结果返回给只读对象

DataReader。

- ExecuteScalar()方法：用于执行查询操作，但只返回查询结果的第一条记录。

14.2 使用 GridView 控件更新数据表

在网页中，能否允许对数据表进行编辑，取决于以下几方面。

- 是否允许访问包括数据表的网页。这个问题将在第 18 章中讲述。
- 数据库和表是否给操作者赋予了编辑的权限。
- 在被编辑的数据表中是否确定了关键字。

只有上述条件全部满足时，才能对数据表进行编辑。

修改(更新)数据表的具体步骤如下。

(1) 网页中放入 GridView 控件。通过数据源控件与数据库连接。

(2) 当选择好数据表以及相关字段以后，单击【高级】按钮，如图 14.1 所示。

这里提供了两个复选框。选中第一个复选框时，系统将自动产生增加(Insert)、更新(Update)和删除(Delete)的 SQL 语句；选择第二个复选框时，有助于防止由于同时对数据表进行更新和删除而并发的冲突。

(3) 回到 GridView 控件并打开数据源控件，在弹出的对话框中将看到【启用编辑】与【启用删除】等选项，如果选中这些复选框，系统将在 GridView 控件中显示出相应的按钮(如【编辑】、【删除】按钮等)，如图 14.2 所示。

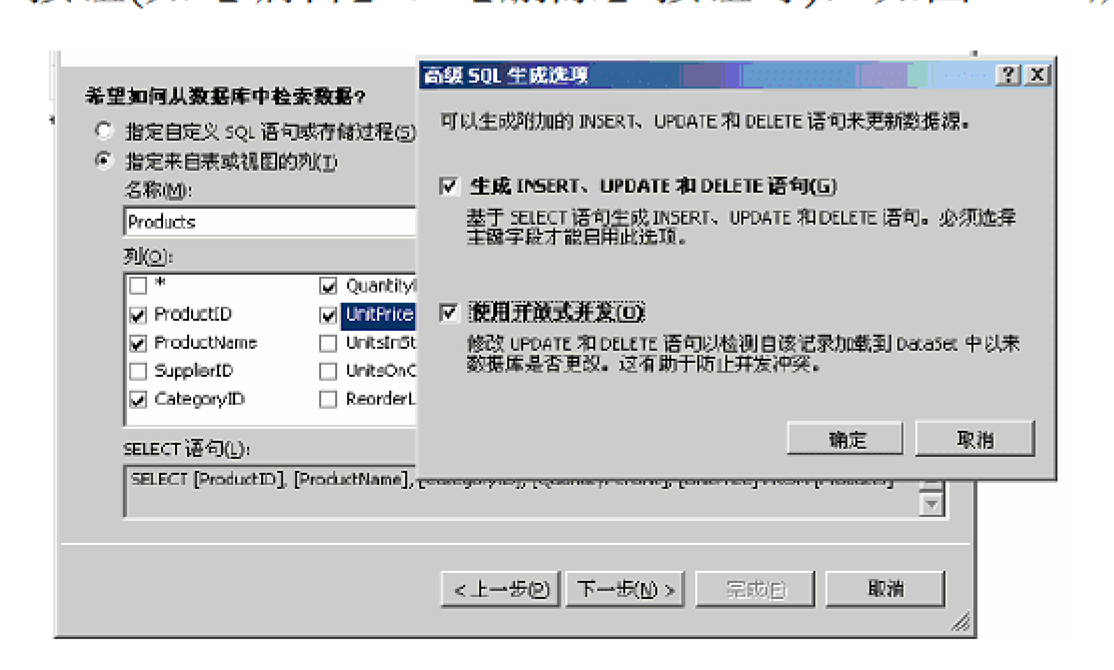


图 14.1 通过【高级】按钮设置 SQL 编辑语句　　　　图 14.2 启用编辑和删除选择

现在打开【源】视图，就可以看到系统不仅已经生成了编辑数据表的 SQL 语句，同时还生成了参数赋值的语句。代码如下。

```
<asp:SqlDataSource ID="SqlDataSource1" Runat="server"
DeleteCommand=
<!--删除命令-->
  "DELETE FROM [Products] WHERE [ProductID] = @original_ProductID AND
  [ProductName] = @original_ProductName AND [CategoryID] =
  @original_CategoryID AND [QuantityPerUnit] = @original_QuantityPerUnit
  AND [UnitPrice] =  @original_UnitPrice"
<!--增加命令-->
 InsertCommand="INSERT INTO [Products] ([ProductName], [CategoryID],
```

```
        [QuantityPerUnit], [UnitPrice]) VALUES (@ProductName, @CategoryID,
        @QuantityPerUnit, @UnitPrice)"
    <!--显示命令-->
     SelectCommand="SELECT [ProductID], [ProductName], [CategoryID],
        [QuantityPerUnit], [UnitPrice] FROM [Products]"
    <!--更新命令-->
    UpdateCommand="UPDATE [Products] SET [ProductName] = @ProductName,
        [CategoryID] = @CategoryID, [QuantityPerUnit] = @QuantityPerUnit,
        [UnitPrice] = @UnitPrice WHERE [ProductID] = @original_ProductID AND
        [ProductName] = @original_ProductName AND [CategoryID] =
        @original_CategoryID AND [QuantityPerUnit] =
        @original_QuantityPerUnit AND [UnitPrice] = @original_UnitPrice"

    ConnectionString="<%$ ConnectionStrings:AppConnectionString1 %>"
    ConflictDetection="CompareAllValues">
    <!--删除命令中的参数-->
    <DeleteParameters>
       <asp:Parameter Type="Int32" Name="ProductID"></asp:Parameter>
       <asp:Parameter Type="String" Name="ProductName"></asp:Parameter>
       <asp:Parameter Type="Int32" Name="CategoryID"></asp:Parameter>
       <asp:Parameter Type="String" Name="QuantityPerUnit"></asp:Parameter>
       <asp:Parameter Type="Decimal" Name="UnitPrice"></asp:Parameter>
    </DeleteParameters>
    <!--更新命令中的参数-->
    <UpdateParameters>
       <asp:Parameter Type="String" Name="ProductName"></asp:Parameter>
       <asp:Parameter Type="Int32" Name="CategoryID"></asp:Parameter>
       <asp:Parameter Type="String" Name="QuantityPerUnit"></asp:Parameter>
       <asp:Parameter Type="Decimal" Name="UnitPrice"></asp:Parameter>
       <asp:Parameter Type="Int32" Name="ProductID"></asp:Parameter>
    </UpdateParameters>
    <!--增加命令中的参数-->
    <InsertParameters>
       <asp:Parameter Type="String" Name="ProductName"></asp:Parameter>
       <asp:Parameter Type="Int32" Name="CategoryID"></asp:Parameter>
       <asp:Parameter Type="String" Name="QuantityPerUnit"></asp:Parameter>
       <asp:Parameter Type="Decimal" Name="UnitPrice"></asp:Parameter>
    </InsertParameters>
  </asp:SqlDataSource>
```

(4) 运行程序后单击【编辑】按钮时，各个字段的值上出现 TextBox 控件，以便填入新数据。原来的【编辑】按钮此时也变成了【更新】和【取消】两个按钮，如图 14.3 所示。

编辑	ProductID	ProductName	CategoryID	QuantityPerUnit	UnitPrice
编辑 删除 选择	1	苹果汁	1	每箱24瓶	19.0000
编辑 删除 选择	2	蕃茄酱	1	每箱24瓶	8.9230
更新 取消	3	盐	2	每箱24瓶	5.5500
编辑 删除 选择	9	龙虾	8	每箱24瓶	48.0000
编辑 删除 选择	10	味精	2	每箱24瓶	23.0000
1 2 3 4 5 6 7 8					

图 14.3　编辑字段

(5) 在各个 TextBox 控件中填完新数据后，若单击【更新】按钮将完成更新工作；若单击【取消】按钮，则此次的更新作废，恢复原状。

如果只希望修改某些字段而不是修改全部字段时，还需另外修改或增加一些设置。例如只允许更新“单价”字段，显示的界面如图 14.4 所示。为了实现只对“单价”的更新，需要增加以下设置。



编辑	产品ID	产品名	类型ID	单元数量	单价
编辑 删除 选择	1	苹果汁	1	每箱24瓶	19.0000
修改 作废	2	蕃茄酱	1	每箱24瓶	8.9230
编辑 删除 选择	3	盐	2	每箱24瓶	5.5500
1 2 3 4 5 6 7 8 9 10 ...					

图 14.4 编辑部分字段

① 在字段属性中将除单价(UnitPrice)以外的各字段的 ReadOnly 属性设置为 True。

② 打开【源】视图，修改相关的 SQL 命令。修改后的更新语句如下。

```
UpdateCommand="UPDATE [Products] SET [UnitPrice] = @UnitPrice
                WHERE [ProductID] = @original_ProductID "
```

14.3 使用 GridView 控件的列模板

通过对 GridView 控件中列(Columns)属性的设置，可以改变控件显示的格式，还可以增添新功能。例如，可以在 GridView 控件中增加按钮；为字段的更新增加校验功能；显示各记录的图片等。下面将分别讲述这些功能的实现方法。

默认情况下，GridView 控件总是按照数据源的结构显示数据。例如，数据源中的数据表(DataTable)中包含 4 个字段，20 条记录。在 GridView 的对象中也将显示同样的字段和记录，而且表中的字段名与数据源的名字也相同。

14.3.1 选择显示的字段

若想改变列的显示格式时，可按以下方法进行。

(1) 打开 GridView 控件的【属性】对话框，单击【列】属性右边的省略号按钮，弹出如图 14.5 所示的对话框。

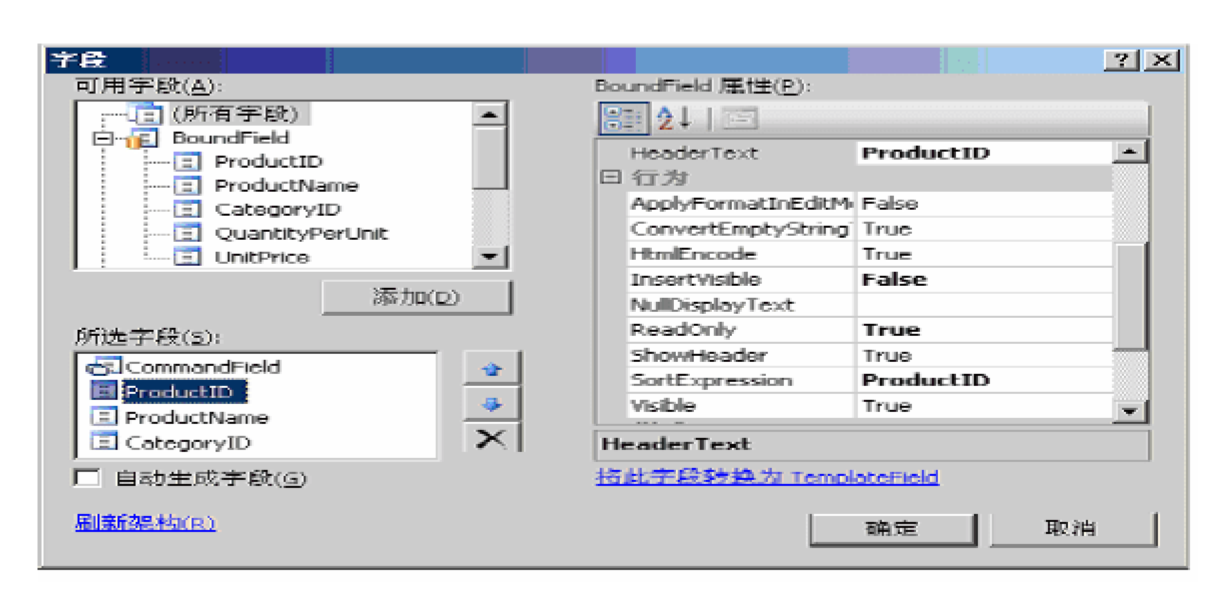


图 14.5 设置 GridView 的列属性

(2) 取消选中【自动生成字段】复选框。如果不取消选中这个复选框，当按照下面的方法增加新字段时，GridView 控件中将会出现重复的字段。

(3) 在【所选字段】列表框中逐个选择字段，并在右边【行为】卷展栏中分别将列标题(HeaderText)改用中文标题。

14.3.2　增添按钮

展开【可用字段】列表框中的 CommandField，下面显示【编辑】、【更新】、【取消】、【选择】、【删除】几组按钮。根据需要可以将这些按钮添加到下面的【所选字段】列表框中。图 14.6 就是将【编辑】、【更新】、【取消】按钮增加进来后的情况。

单击这个新增加的按钮，然后在右上方的列表框中设置属性。例如填写栏名(HeaderText)，将英文显示改用汉字。

14.3.3　使用模板列

当将某列转换成模板列时，就意味着可以为该列设置多种不同的状态(例如被选择状态、编辑状态等)，并为不同的状态增添控件和方法。

转换的方法是，先选择某一列名，然后单击对话框右下角的【将此字段转换为 TemplateField】，在图 14.6 所示的对话框中即可将该列转换成模板列，然后在模板列的不同状态中增加控件和方法。

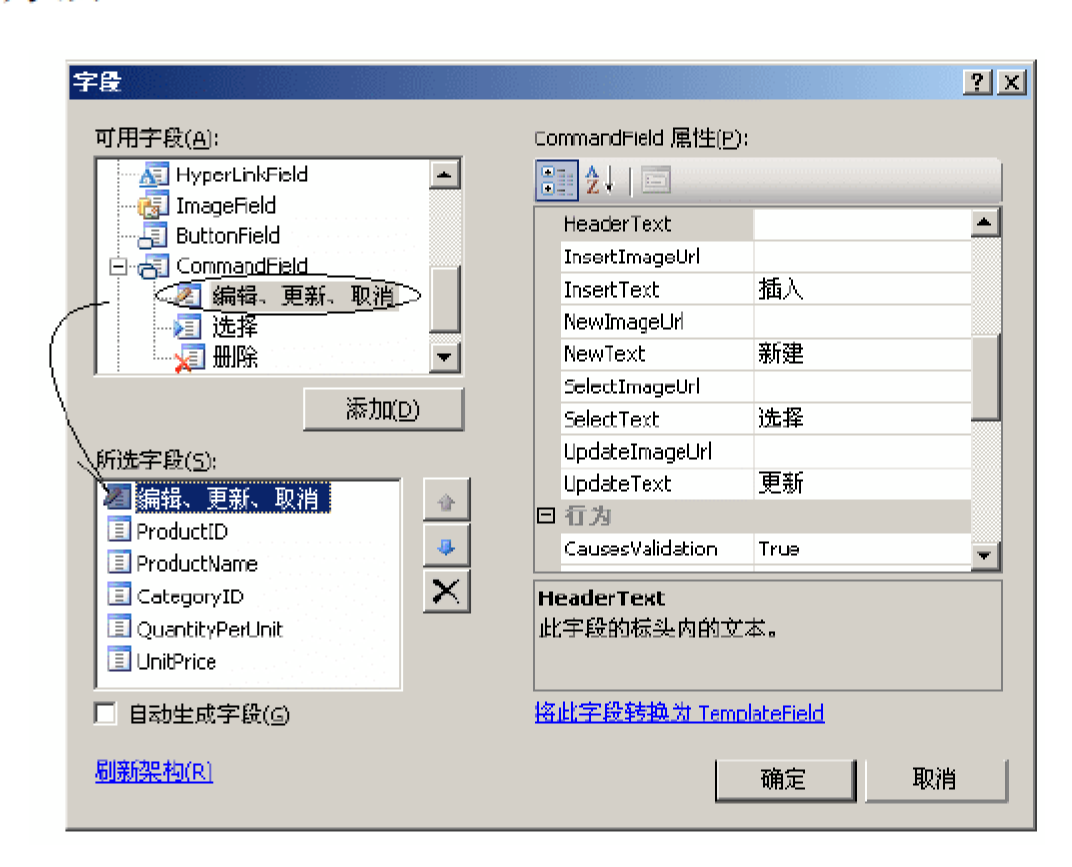
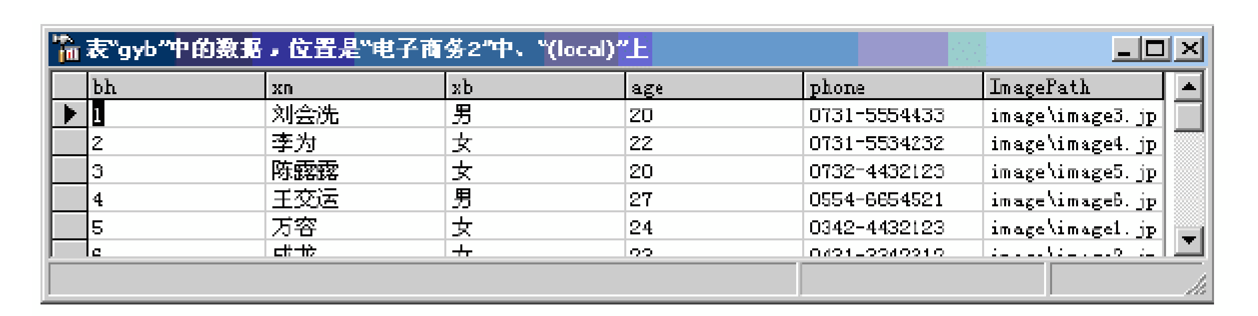


图 14.6　增添按钮

下面利用一张雇员表 (gyb)来说明设置方法，表格的结构与内容如图 14.7 所示。


表"gyb"中的数据，位置是"电子商务2"中、"(local)"上

bh	xn	xb	age	phone	ImagePath
1	刘会洗	男	20	0731-5554433	image\image3. jp
2	李为	女	22	0731-5534232	image\image4. jp
3	陈露露	女	20	0732-4432123	image\image5. jp
4	王交运	男	27	0554-6654521	image\image6. jp
5	万容	女	24	0342-4432123	image\image1. jp

图 14.7　雇员表

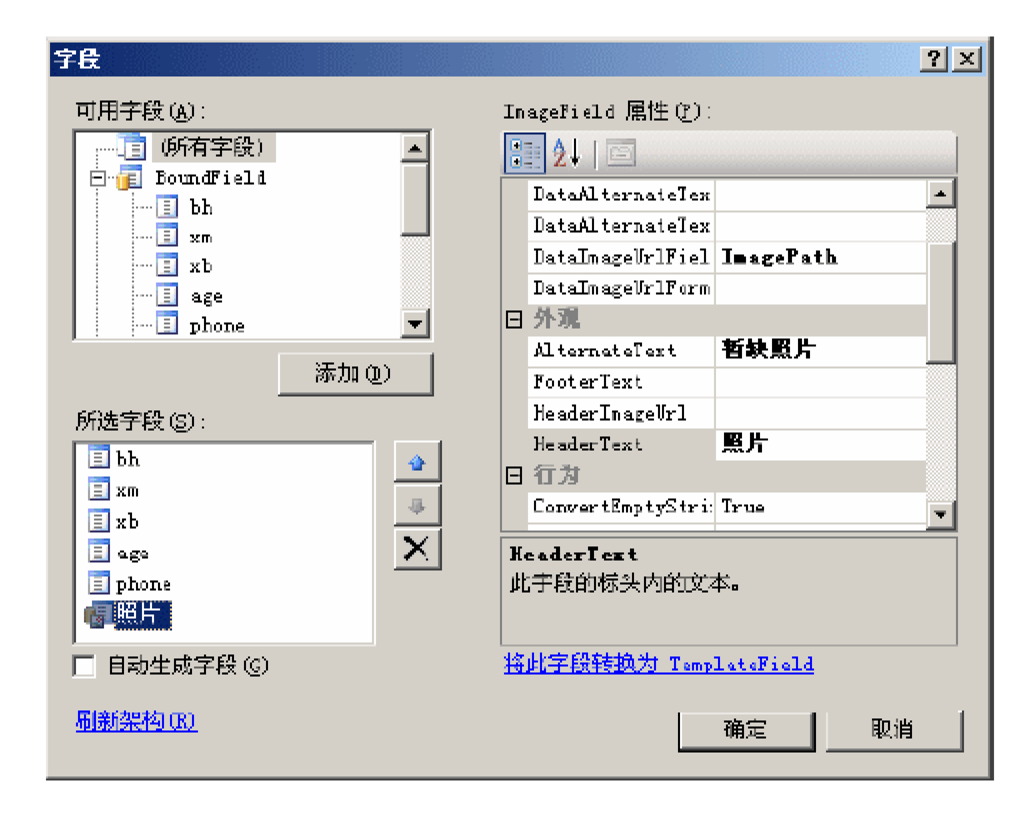
1. 准备工作

(1) 在窗体页中放入 GridView 控件，建立与数据库的连接。

(2) 通过 GridView 控件的 Columns 属性设置模板，并分别将各字段(关键字 bh 字段例外)转换为 TemplateField，如图 14.8 所示。

2. 为姓名字段设置模板

(1) 右击 GridView 控件右上方的小箭头[>]，并在弹出的窗口中选择【编辑模板】命令，将弹出各模板的编辑模式列，情况如图 14.9 所示。



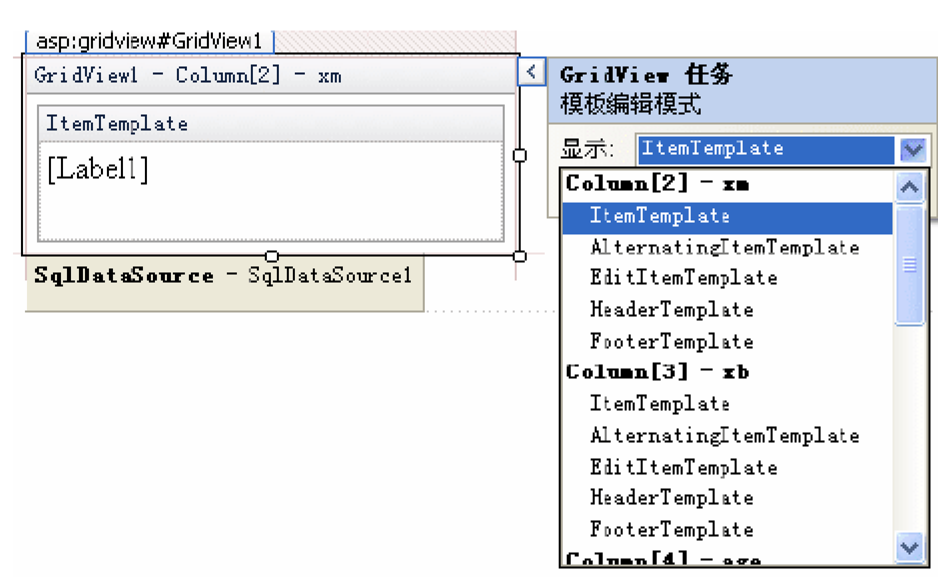


图 14.8 设置模板　　　　图 14.9 模板的编辑模式

(2) 选择 Column[2] - xm 的 ItemTemplate 模板，在弹出的窗口中[Label1]控件的右方增加校验控件(RequiredFieldValidator)。选择编辑 DataBindings 数据绑定项，按照图 14.10 的界面进行设置。

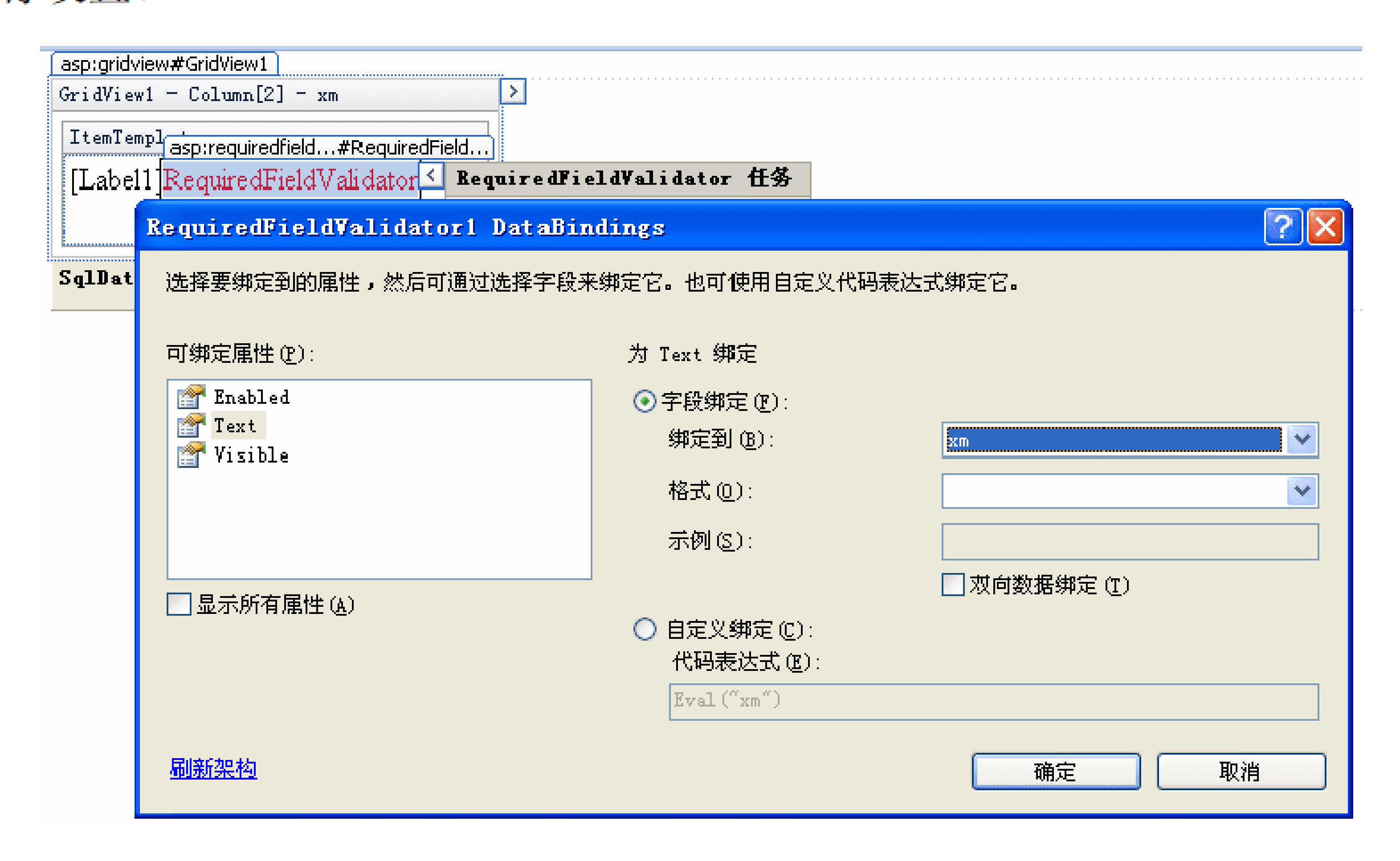


图 14.10 ItemTemplat 模板的数据绑定界面

(3) 给 RequiredFieldValidator 控件设置其他属性。

此时控件的 Text 与 xm 字段进行单向绑定，在代码表达式中将显示出 Eval("xm")。实

际的数据绑定语句是：

```
Text='<%# Eval("xm") %>'
```

(4)　选择 EditItemTemplate 模板，并在模板的 TextBox 控件右端放入 RequireValidator 控件，并将校验对象指向 TextBox，并且按照要求写出当出现错误时的提示信息。

(5)　将控件的 Text 与 xm 字段进行双向绑定，情况如图 14.11 所示。在代码表达式中将显示出 Bind("xm")。实际的数据绑定语句是：

```
Text='<%# Bind("xm") %>'
```

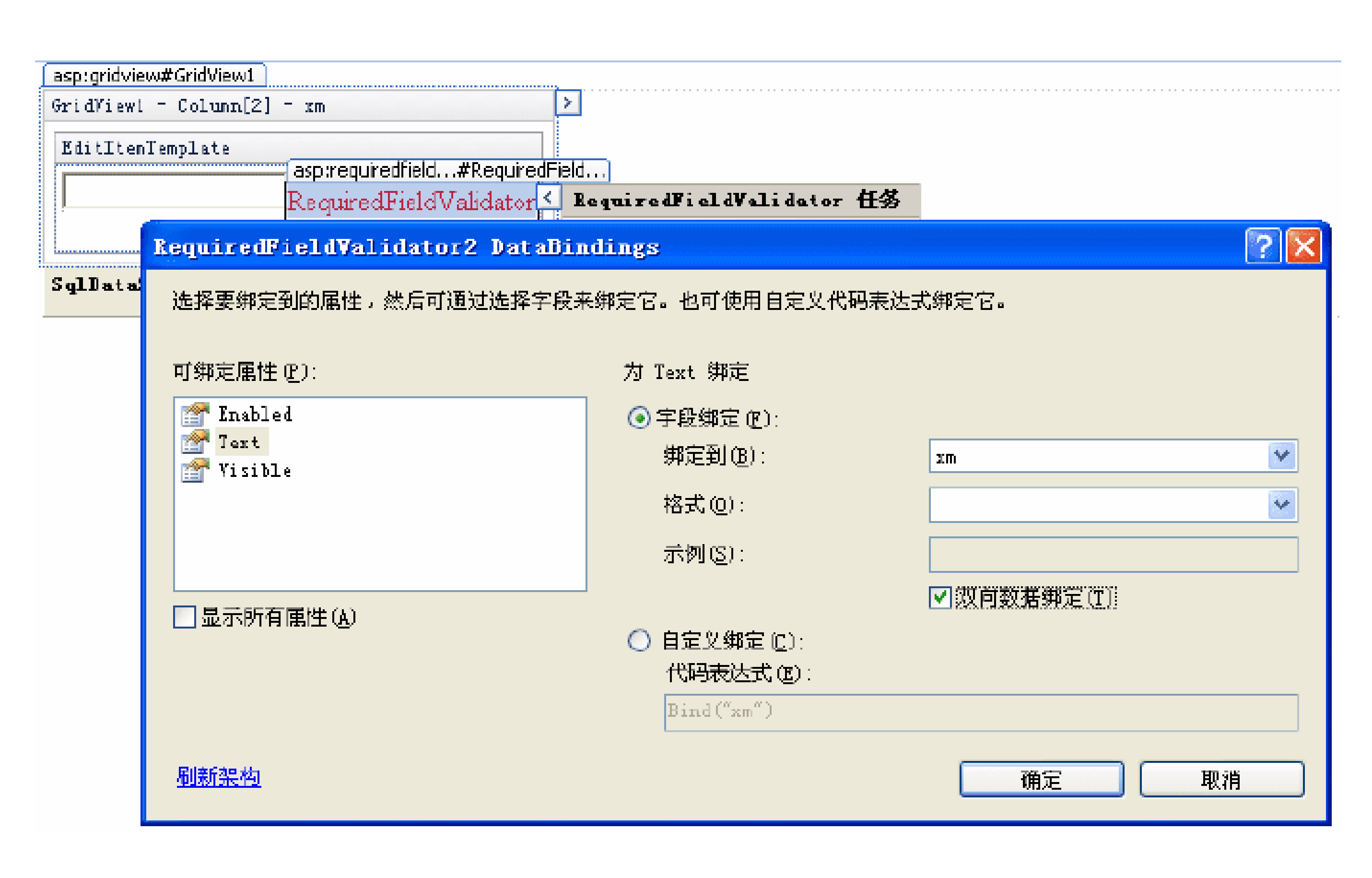


图 14.11　EditItemTemplat 模板的数据绑定

3. 为性别字段设置模板

在图 14.9 所示的菜单中选择【Column[3]-xb】，然后在编辑模板中增加 RadioButtonList 和 RequiredValidator 控件，对性别的编辑进行校验。情况如图 14.12 所示，其方法与对姓名的设置基本相同，这里不再重复。

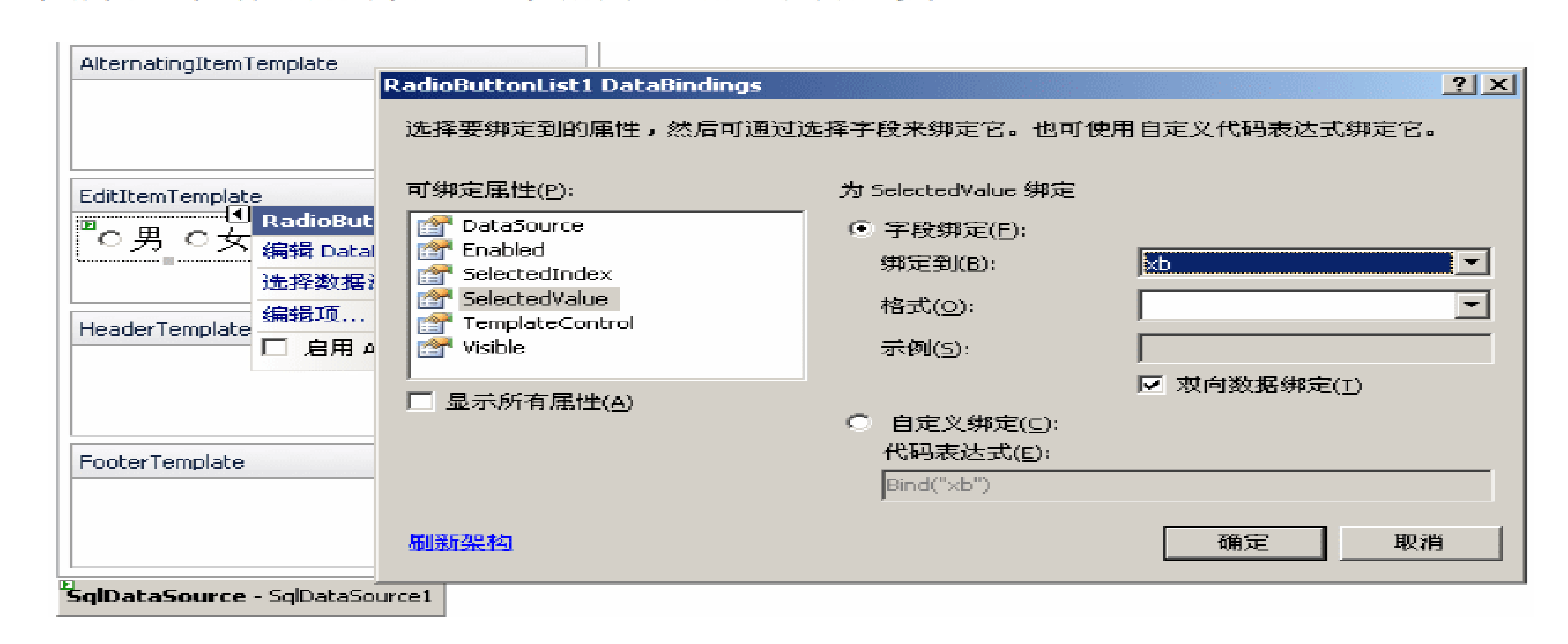


图 14.12　给年龄的输入增加校验功能

4. 为年龄字段设置模板

(1)　在图 14.9 所示的菜单中选择【Column[4]-age】。

(2) 在 ItemTemplate 模板中放入 Label 控件，右击该控件后，选择【属性】命令，然后将其 ID 属性取名为 Label3。

(3) 在 EditItemTemplate 模板中放入 TextBox 及 RangeValidator 控件，将其 MaxmumValue 属性设置为 65，将其 MinimumValue 属性设置为 18，将类型设置为 Integer，将校验对象指向 TextBox，并且按照要求改写出现错误时的提示信息。

(4) 按照前面讲述的方法将输入窗口的 Text 属性绑定在 age 字段上。

此时在【源】视图中模板的代码如下。

```
<asp:TemplateField SortExpression="age" HeaderText="年龄">
  <EditItemTemplate>
    <asp:TextBox Runat="server" Text='<%# Bind("age") %>' ID="TextBox3"
         Width="60px"></asp:TextBox>
    <asp:RangeValidator ID="RangeValidator1" Runat="server"
         ErrorMessage="年龄需在 18 岁到 65 岁之间"
         ControlToValidate="TextBox3" MaximumValue="65" MinimumValue="18"
         Type="Integer">
    </asp:RangeValidator>
  </EditItemTemplate>
  <ItemTemplate>
    <asp:Label    Runat="server"    Text='<%#    Bind("age")    %>'
         ID="Label3"></asp:Label>
  </ItemTemplate>
</asp:TemplateField>
```

5. 为照片字段设置模板

这里介绍的是另一种显示图像的方法，即通过模板显示图像。具体步骤如下。

(1) 在图 14.9 所示的菜单中选择【Column[4]-age】。

(2) 在 ItemTemplate 模板中放入 Image 控件，右击该控件后，选择【属性】命令，然后将其 AlternateText 属性设置为“没有提供图片”。

(3) 按照前面讲述的方法实施数据绑定，将 ImageUrl 属性与 ImagePath 字段绑定，如图 14.13 所示。此时将出现以下数据绑定的语句。

```
ImageUrl='<%# Bind("ImagePath") %>'
```

此时在【源】视图中看到模板的代码如下。

```
<asp:TemplateField SortExpression="ImagePath" HeaderText="照片">
  <EditItemTemplate>
    <asp:TextBox   Runat="server"   Text='<%#   Bind("ImagePath")   %>'
         ID="TextBox4">
    </asp:TextBox>
  </EditItemTemplate>
  <ItemTemplate>
    <asp:Image ID="Image1" Runat="server" ImageUrl='<%# Bind("ImagePath")
         %>' />
  </ItemTemplate>
</asp:TemplateField>
```

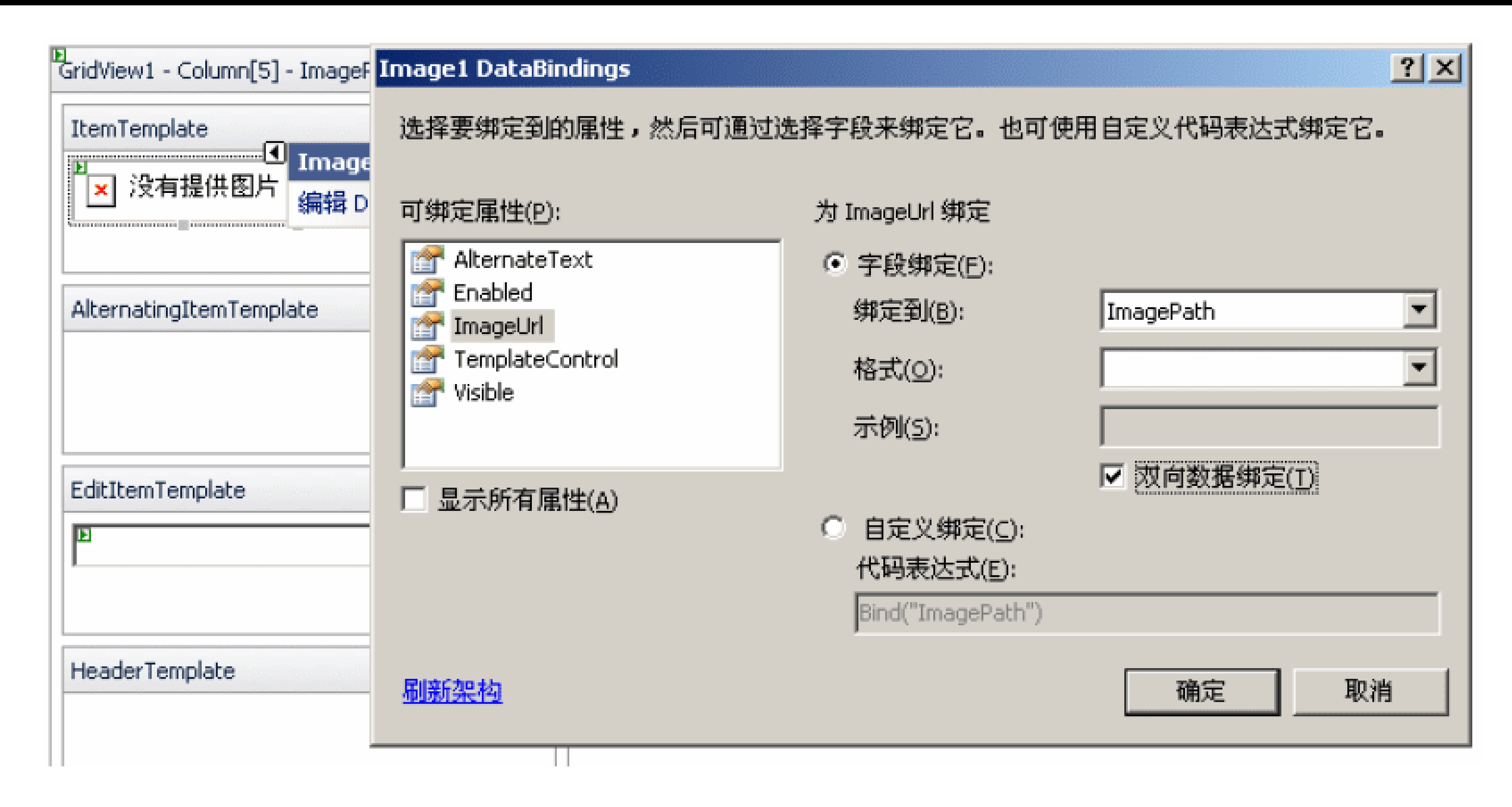


图 14.13　编辑照片模板

(4) 经过上面的设置后运行的界面如图 14.14 所示。



选择	编号	姓名	性别	年龄	电话	照片
选择	2211	成工	男	22	0731-55566611	
选择	2212	刘归	女	20	0732-44667733	
1 2 3						

图 14.14　通过模板显示图像

其中在编辑模板中的设置只有在编辑数据时才会显示出来。

14.4　使用 GridView 控件增添记录

如前所述，在配置 GridView 控件的数据源控件时单击【高级】按钮就会自动生成 Insert、Update 和 Delete 的 SQL 编辑语句，并且给出各种参数类型的语句。其中有关 Insert 的 SQL 语句如下。

```
InsertCommand="INSERT  INTO  [gyb]  ([xm],  [xb],  [age],  [phone],
[ImagePath])
VALUES (@xm, @xb, @age, @phone, @ImagePath)"
```

相应的参数类型语句如下。

```
<InsertParameters>
   <asp:Parameter Type="String" Name="xm"></asp:Parameter>
   <asp:Parameter Type="String" Name="xb"></asp:Parameter>
   <asp:Parameter Type="Int32" Name="age"></asp:Parameter>
   <asp:Parameter Type="String" Name="phone"></asp:Parameter>
   <asp:Parameter Type="String" Name="ImagePath"></asp:Parameter>
</InsertParameters>
```

如果需要在数据表中增添新记录，则需要在前面步骤的基础上，再补充以下步骤。

(1) 在窗体中增添几个输入框(例如增加了 5 个 TextBox)。

(2) 在 GridView 控件中增添一个按钮，并将该按钮的 CommandName 属性命名为 insert。

(3) 双击 RowCommand 事件，书写代码给待定参数赋值。代码如下。

```
void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        if (e.CommandName == "insert")
        {
            SqlDataSource1.InsertParameters.Clear();
            SqlDataSource1.InsertParameters.Add("xm",TextBox1.Text);
            SqlDataSource1.InsertParameters.Add("xb", TextBox2.Text);
            SqlDataSource1.InsertParameters.Add("age", TextBox3.Text);
            SqlDataSource1.InsertParameters.Add("phone", TextBox4.Text);
            SqlDataSource1.InsertParameters.Add("ImagePath",
                TextBox5.Text);
            SqlDataSource1.Insert();
        }
    }
```

这里的语句中参数都采用 string 类型。即

```
SqlDataSource1.InsertParameters.Add(string, string);
```

注意：数据表中的 bh 字段是自动增加的关键字段，增加新记录时不需要赋值。

14.5 使用 DetailsView 控件

DetailsView 是 ASP.NET 3.5 提供的一个新控件，和 GridView 控件一样都继承于类库中的 CompositeDataBoundControl 类，因此它们之间有很多共性。DetailsView 控件也可以通过数据源控件连接到数据库，并且也能够对数据表的记录进行插入、编辑或删除操作。与 GridView 控件的最大不同点是，GridView 是一个面向记录集合的控件，而 DetailsView 是一个面向单条记录的控件。在 DetailsView 控件的界面中每次只显示一条记录，而且内容按照垂直方式进行排列。在查询中如果有多条记录符合要求时，可以事先利用类似于设置分页的方法进行设置，即将 DailsView 控件的 AllowPaging(允许分页)属性设为 true，然后逐条记录浏览结果。

将 DatailsView 控件结合 GridView 或者 DropDownList 使用时，该控件非常适合于作为子表进行同步。

利用 DetailsView 控件增添(Insert)记录时特别方便，因为它不需要另外增加输入框。

现在以 Category(父表)和 Products(子表)同步为例，用 GridView 显示父表，用 DetailsView 来显示子表。两表同步的结果如图 14.15 所示。代码如下。



图 14.15 利用 DetailsView 控件

```
<%@ Page Language="C#" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<script runat="server">
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
<!--下面是 GridView 部分(父表)-->
  <asp:GridView ID="GridView1" Runat="server
        DataSourceID="SqlDataSource1" DataKeyNames="CategoryID"
        AutoGenerateColumns="False" AllowPaging="True"
        PageSize="3">
    <Columns>
    <asp:CommandField HeaderText="选择" ShowSelectButton="True"
         SelectText="选择">
    </asp:CommandField>
    <asp:BoundField ReadOnly="True" HeaderText="类型 ID"
         InsertVisible="False" DataField="CategoryID" SortExpression="CategoryID">
    </asp:BoundField>
    <asp:BoundField HeaderText="类型名" DataField="CategoryName"
         SortExpression="CategoryName">
    </asp:BoundField>
    </Columns>
 </asp:GridView>
 <!— 生成 GridsView 的数据源控件 →
 <asp:SqlDataSource ID="SqlDataSource1" Runat="server"
   SelectCommand="SELECT [CategoryID], [CategoryName] FROM [Categories]"
   ConnectionString="<%$ ConnectionStrings:AppConnectionString1 %>">
 </asp:SqlDataSource>
<!--下面是 DetailView(子表)部分-->
<asp:DetailsView  ID="DetailsView1"  Runat="server"  DataSourceID="SqlDataSource2"
     DataKeyNames="ProductID"
    AutoGenerateRows="False" AllowPaging="True" >
   <Fields>
    <asp:BoundField ReadOnly="True" HeaderText="产品 ID"
           InsertVisible="False" DataField="ProductID"
           SortExpression="ProductID">
     </asp:BoundField>
    <asp:BoundField  HeaderText=" 产 品 名 "  DataField="ProductName"  SortExpression=
         "ProductName">
     </asp:BoundField>
    <asp:BoundField  HeaderText=" 类 型  ID"  DataField="CategoryID"  SortExpression=
         "CategoryID">
     </asp:BoundField>
    <asp:BoundField HeaderText="单元数量" DataField="QuantityPerUnit"
         SortExpression="QuantityPerUnit">
```

```
        </asp:BoundField>
       <asp:BoundField HeaderText="单价" DataField="UnitPrice" SortExpression=
             "UnitPrice">
       </asp:BoundField>
     </Fields>
   </asp:DetailsView>
   <!-- 生成DetailsView 的数据源控件-->
   <asp:SqlDataSource ID="SqlDataSource2" Runat="server" SelectCommand="SELECT
   [ProductID], [ProductName], [CategoryID],
                 [QuantityPerUnit], [UnitPrice] FROM [Products]
                 WHERE ([CategoryID] = @CategoryID)"
   ConnectionString="<%$ ConnectionStrings:AppConnectionString1 %>">
   <!--自动生成参数的类型-->
     <SelectParameters>
      <asp:ControlParameter Name="CategoryID" DefaultValue="1" Type="Int32"
            ControlID="GridView1" PropertyName="SelectedValue">
      </asp:ControlParameter>
    </SelectParameters>
  </asp:SqlDataSource>
 </div>
 </form>
 </body>
 </html>
```

代码中已经分别说明了它们的作用。总的来说，代码的前半部分是有关 GridView 的代码，后半部分是 DetailsView 的代码。每个显示控件使用自己的数据源控件。在 GridView 控件中使用了选择按钮。在 DetailsView 控件中用查询方法取得同步，但每次只显示一条记录，如果将 DetailsView 控件的 AllowPaging 属性设为 True，在控件的下面将出现分页标志，利用这些标志可以查看其他同步记录。

下面说明利用 DetailsView 控件进行插入记录的操作，这一点比 GridView 方便得多。当通过数据源控件设置了“编辑”操作后，控件下方会出现 Edit、Delete、New 编辑按钮。若单击 New(插入)按钮时将弹出新的界面，界面中显示除关键字(产品 ID)以外的输入框，此时的 New 按钮变成了 Insert 和 Cancel 两个按钮。在输入框中填入新数据后若单击 Insert 按钮就完成了插入新记录的工作。若单击 Cancel 按钮则将插入记录的操作作废。界面如图 14.16 所示。

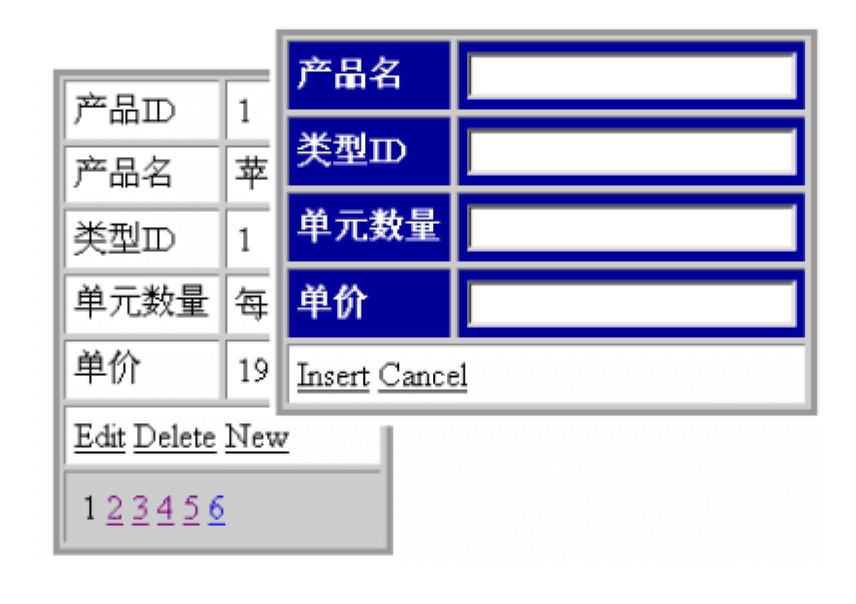


图 14.16　利用 DetailsView 控件进行编辑

14.6　小　　结

数据表的编辑是利用 SQL 语句进行的。在配置数据源控件的过程中，若单击【高级】按钮，系统就会自动生成 SQL 的增、删、改的语句，以及参数类型的语句。更新数据时只需要在弹出的输入框中填入数据，然后单击【更新】按钮。实际应用中常常只允许更新部分字段的数据。为了实现这一功能需要做两方面的工作：在 GridView 控件的字段编辑中将不允许更新的字段的 ReadOnly 属性设为 true；在 SQL 的 Update 命令中删除不允许更新的字段。

模板在 GridView 控件中具有很强的功能，在这里可以增加编辑时的校验功能以及其他功能。

DetailsView 控件是基于单条记录的控件，而且通常按照垂直方式来显示记录。DetailsView 可以单独使用，也可以与其他控件配合使用。当与其他控件配合使用时非常适合于承担子表的任务。利用 DetailsView 编辑记录很方便，特别是用于增添记录时，会自动弹出输入框而不需要附加其他控件。

14.7　习　　题

1. 选择题

(1) 在配置 GridView 控件的 SqlDataSource 数据源控件过程中，单击【高级】按钮的目的是______。

A. 打开其他对话框　　B. 输入新参数

C. 生成 SQL 编辑语句　　D. 优化代码

(2) 在配置 GridView 的 SqlDataSource 数据源控件过程中，单击【高级】按钮后新打开的对话框中的选项显示无效，这常常是因为______。

A. 不能输入参数　　B. 不能返回数据

C. 不能优化代码　　D. 数据表中缺少关键字段

(3) GridView 列模板的作用是________。

A. 增加功能　　B. A+C

C. 改善数据表的显示　　D. 定义列格式

2. 判断题

(1) GridView 是一个面向记录集合的控件，而 DetailsView 是一个面向单条记录的控件。　(　　)

(2) 在 DetailsView 控件中不能显示符合条件的多条记录。　(　　)

(3) 利用 DetailsView 控件增添记录特别方便，因为不需要另外增加输入框。　(　　)

3. 简答题

(1) 写出查询、插入、更新、删除数据表记录的 SQL 基本语句。

(2) 允许对数据表进行编辑的必要条件是什么？

(3) 如果只允许对数据表的部分字段进行修改，应该补充哪些设置？

(4) 简述在 GridView 控件中实现给数据表增添记录的设计步骤。

4. 操作题

(1) 设计一个程序只允许修改 Northwind 样板库中 Products 数据表中的 Price 字段。

(2) 设计一个用 GridView 控件作为父表，DetailsView 控件作为子表的同步程序。同时能够在 DetailsView 控件中对数据表进行编辑(包括增加、删除和修改功能)。

(3) 自行设计一学生基本情况表(包括学号、姓名、性别、年龄、照片等字段)，用 GridView 控件显示该表，并且通过列模板在输入或修改数据时进行合法性验证。

第 15 章　ListView 与 DataPager 控件

ListView 是 ASP.NET 3.5 提供的一个新控件，它综合了原有一些控件，如 GridView、FormView、DataList、Repeater 等的某些特点。在这个新控件中大量使用了模板技术，因而在布局方面具有非常灵活的自定义能力，非常适合于开发那些商品外观十分重要的电子商务，如服装、花卉、首饰等网上商店。

DataPager 是一个分页控件，放在 ListView 中帮助进行分页工作。

在本章中，将先介绍控件中使用的模板及其作用，然后讲述利用这些模板绑定数据的方法。具体问题包括：

- ListView 控件中的模板。
- 模板中绑定数据的方法。
- 用网格方式显示数据。
- 用平铺方式显示数据。

15.1　ListView 控件中的模板

ListView 通过模板进行布局是它的最大特点和优点。因此了解这些模板的作用以及它们之间的关系很有必要。ListView 控件中使用的模板有如下几个。

- ItemTemplate：绑定数据的主模板。用来定义并显示各数据的绑定项。
- AlternatingItemTemplate：在连续的记录中区别交替记录的模板。通常情况下，交替记录的模板内容和样式相同，只是底色有所区别。
- SelectedItemTemplate：被选中记录的模板。
- EmptyItemTemplate：空记录模板。当被绑定的数据为空时显示的模板。
- ItemSeparatorTemplate：数据绑定项之间显示内容的模板。
- GroupTemplate：分组布局模板。
- GroupSeparatorTemplate：每组数据绑定之间显示内容的模板。
- EditItemTemplate：编辑记录模板。此模板与 ItemTemplate 模板相比，主要有两个区别。
    - ◆ 用 TextBox 控件代替 Label 控件。
    - ◆ 用<%# Bind("…") %>数据绑定语句代替<%# Eval("…") %>语句。
- InsertItemTemplate：插入新记录模板。其特点与 EditItemTemplate 相同。
- LayoutTemplate：用于布局的根模板。模板中只包括标签(如<table>、<tr>、<td>等)和占位符。其占位符的位置将被其他模板所取代。代码如下：

```
<table ID="groupPlaceholderContainer" runat="server"…>
```

表示这里的占位符将被分组模板 GroupTemplate 来取代。

```
<table ID="itemPlaceholderContainer" runat="server"…>
```

表示这里的占位符将被主模板 ItemTemplate 来取代。

在此模板中还可以包括一个 DataPager 控件，用来进行分页。

控件中，除 LayoutTemplate 与 ItemTemplate 是必需的以外，其他模板都是可以选择的。

15.2 模板中绑定数据的方法

在模板中进行数据绑定，需要用到系统提供的 Eval(“字段名”)或者 Bind(“字段名”)方法。Eval()是一个静态方法，不管字段中是什么数据类型，它总是返回字符串，以便在网页中显示，使用时不必关心数据本来的类型以及如何转换。Eval()只能用于数据显示控件的模板中，Eval()方法必须写在“<%#...%>”的标签中。

Bind()也是一个静态方法，与 Eval()相似，它们都可以从数据源中检索数据并自动转换为字符串。不同的是，Bind()还支持双向绑定。所谓双向绑定，就是除了从数据源获取数据外，还允许客户编辑、删除或插入数据。因此，如果希望实现双向绑定时，就应该使用 Bind()而不使用 Eval()方法。

例如：在连接数据表的基础上使用<%# Eval("ProductName") %> 语句将返回数据表中 ProductName 字段的数据；使用<%# Bind("ProductName") %> 语句也将返回数据表中 ProductName 字段的数据。前一条语句只能用于显示,后一条语句既能显示，又能用于编辑。

15.3 用网格方式显示数据

15.3.1 设计步骤

具体操作步骤如下。

(1) 创建数据库及数据表，并将其放置于 App_Data 目录下。假定数据表的情况如表 15.1 所示。

表 15.1 数据表

	bh	name	age	address	image
	7	伍国庆	32	湖北	images\p7.jpg
	8	刘礼花	24	湖南	images\p8.jpg
	9	伍来红	22	湖北	images\p9.jpg
	10	赵有	27	安徽	images\p10.jpg

(2) 在网页中拖入 ListView 控件，通过数据源控件(SqlDataSource)连接到数据表，连接方法与 GridView 控件相同。

(3) 在 ListView 任务对话框中选择【配置 ListView】项，打开【配置 ListView】对话框，如图 15.1 所示。

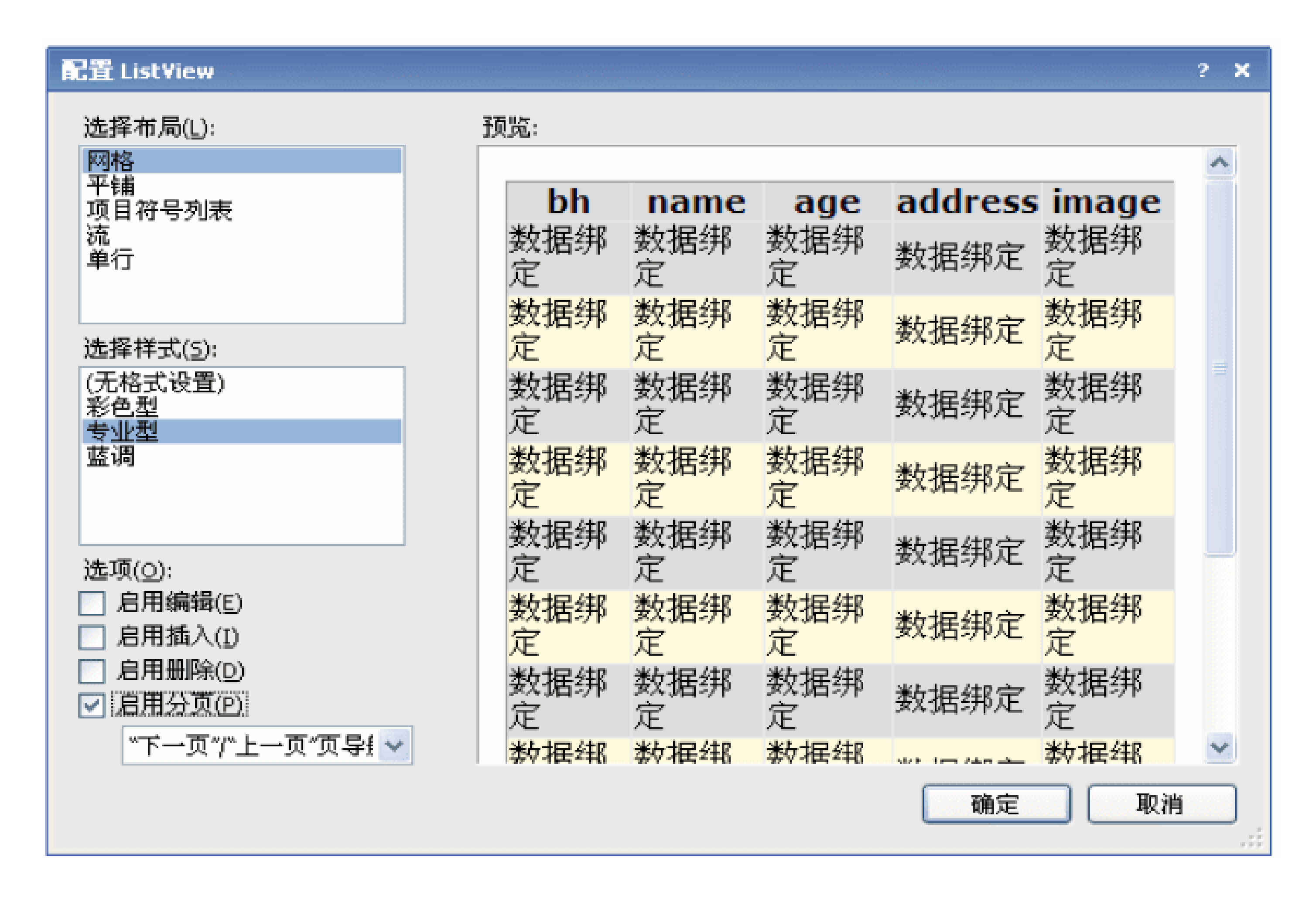


图 15.1　【配置 ListView】对话框

窗口左边包括【选择布局】、【选择样式】和【选项】三部分。每部分的作用如下。

- 【选择布局】列表框包括如下选项。
  - 网格：以表格方式显示数据，其中第一行是标题，后面是各行的记录。
  - 平铺：用“组模板”的平铺布局显示数据，默认时为 3 列(即 GroupItemCount="3"，可修改)。
  - 项目符号列表：数据显示在项目符号列表中，每项包括所有的字段的名称和值。
  - 流：以 div 元素的流布局逐个显示。每项包括所有的字段名称和值。
  - 单行：所有数据都显示在一行的表中。
- 【选择样式】列表框包括 4 项选择，每当选择其中一项时，右边的列表框中将显示出该项选择的样式。
- 【选项】选项组中的【启用编辑】、【启用插入】和【启用删除】等项，只有在配置数据源控件时单击了【高级】按钮(生成了相应的 SQL 语句)时，才能够选择，其使用方法与 GridView 控件相同。选中【启用分页】复选框时，系统将自动在 ListView 控件的 LayoutTemplate 模板中添加一个 DataPager 控件，以实现分页功能。然后再在下面的下拉列表框中进一步确定导航的方法。

15.3.2　模板代码的分析

如果按照图 15.1 所示选择了【网格】、【专业型】、【启用分页】等项，生成的模板代码如图 15.2 所示。

```
<LayoutTemplate>
  <table runat="server">
    <tr runat="server">
      <td runat="server">
        <table ID="itemPlaceholderContainer" runat="server" border="1"
               style="…">
          <tr runat="server" style="…">
            <th runat="server"> bh</th>
            <th runat="server"> name</th>
            <th runat="server"> age</th>
            <th runat="server"> address</th>
            <th runat="server"> image</th>
          </tr>
          <tr ID="itemPlaceholder" runat="server">
          </tr>
        </table>
      </td>
    </tr>
    <tr runat="server">
      <td runat="server" style="…">
        <asp:DataPager ID="DataPager1" runat="server">
          <Fields>
          <asp:NextPreviousPagerField ButtonType="Button" ShowFirstPageButton="True"
                  ShowLastPageButton="True" />
          </Fields>
        </asp:DataPager>
      </td>
    </tr>
  </table>
</LayoutTemplate>
```

布局根模板

占位符,由ItemTemplate模板取代

分页控件

图 15.2　网格模板的代码

注意： 用表格方式显示数据时，每页包括 10 条记录。

```
<ItemTemplate>
  <tr style="…">
    <td>
      <asp:Label ID="bhLabel" runat="server" Text='<%# Eval("bh") %>' />
    </td>
    <td>
      <asp:Label ID="nameLabel" runat="server" Text='<%# Eval("name") %>' />
    </td>
    <td>
      <asp:Label ID="ageLabel" runat="server" Text='<%# Eval("age") %>' />
    </td>
    <td>
      <asp:Label ID="addressLabel" runat="server" Text='<%#
Eval("address") %>' />
    </td>
    <td>
      <asp:Label ID="imageLabel" runat="server" Text='<%# Eval("image") %>' />
    </td>
  </tr></ItemTemplate>
```

现在若想修改某些代码，例如：

(1) 为了在表格中将标题改用汉字，需在布局模板中将<tr…><th…>标记中的标题改成汉字。修改后的代码如下。

```
<td runat="server">
  <table ID="itemPlaceholderContainer" runat="server" border="1"
      style="…">
     <tr runat="server" style="…">
        <th runat="server"> 编号</th>
        <th runat="server"> 姓名</th>
        <th runat="server"> 年龄</th>
        <th runat="server"> 地址</th>
        <th runat="server"> 图像</th>
      </tr>
      <tr ID="itemPlaceholder" runat="server">
      </tr>
     </table>
</td>
```

(2) 如果想在表格中显示图像，在 ItemTemplate 模板中将图像列中的 Label 改用 Image 控件，并将下列代码

```
<asp:Label ID="imageLabel" runat="server" Text='<%# Eval("image") %>' />
```

改为

```
<asp:Image ID="Image1" runat="server" ImageUrl='<%# Eval("image") %>' />
```

15.3.3　修改后显示的界面

经过前面的修改，最后显示的界面如图 15.3 所示。



图 15.3　用网格显示的结果

15.4　用平铺方式显示数据

15.4.1　设计步骤

具体操作步骤如下。

第(1)步与第(2)步与 15.2 节相同。

(3) 在图 15.1 所示的对话框中，分别选择【平铺】、【专业型】和【启用分页】项。

(4) 在 ListView 任务对话框中的【当前视图】下拉列表框中选择 ItemTemplate 项如图 15.4 所示。

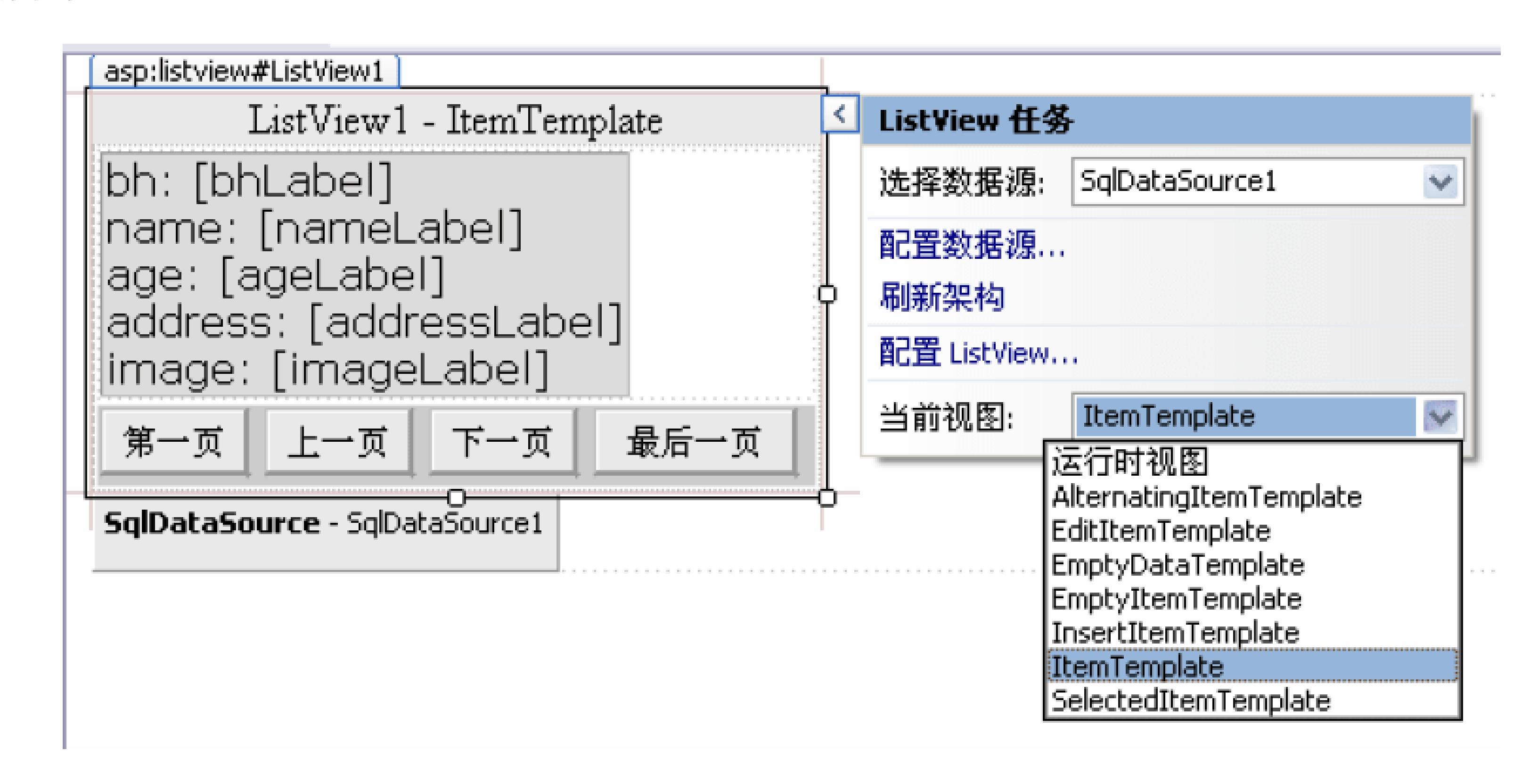


图 15.4　选择 ItemTemplate 项

(5) 在 ItemTemplate 模板中进行布局，并编写数据绑定代码。为此，先去掉<td>标记内的源代码，然后利用表格进行布局，如图 15.5 所示。

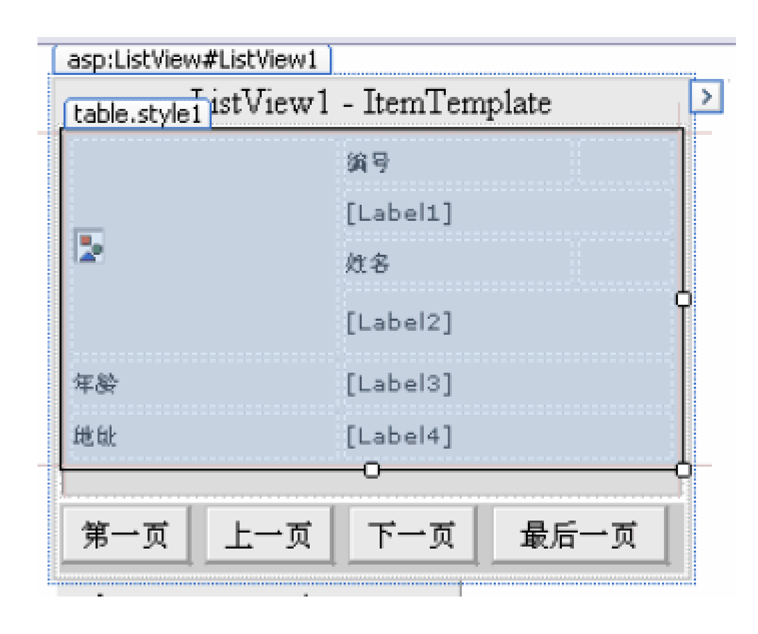


图 15.5　用 ItemTemplate 进行布局

数据绑定的代码如下。

```
<table class="style1">
  <tr>
   <td rowspan="4" class="style2">
    <asp:Image ID="Image1" runat="server"  ImageUrl='<%# Eval("image")
%>'/>
```

```
        </td>
        <td>编号</td>
        <td> </td>
      </tr>
      <tr>
        <td colspan="2">
          <asp:Label ID="Label1" runat="server" Text='<%# Eval("bh")
%>'></asp:Label>
        </td>
      </tr>
      <tr>
        <td>姓名</td>
        <td> </td>
      </tr>
      <tr>
        <td colspan="2" class="style3">
          <asp:Label ID="Label2" runat="server" Text='<%# Eval("name")
%>'></asp:Label>
        </td>
      </tr>
      <tr>
        <td class="style2">年龄</td>
        <td colspan="2">
            <asp:Label ID="Label3" runat="server" Text='<%# Eval("age")
%>'></asp:Label>
        </td>
        </tr>
        <tr>
          <td class="style2">地址</td>
          <td colspan="2">
            <asp:Label ID="Label4" runat="server" Text='<%# Eval("address")
%>'></asp:Label>
          </td>
        </tr>
      </table>
```

(6) 为了在交替记录中显示不同的底色，应将 ItemTemplate 模板中的代码，除 background-color 属性值以外全部复制到 AlternatingItemTemplate 模板下(即原来的 AlternatingItemTemplate 模板中的 background-color 属性值保持不变)。

15.4.2　模板的代码分析

此处的代码和 15.3.2 节中的模板代码基本相同，只是增加了 GroupTemplate 模板。几个模板之间的关系如图 15.6 所示。

可以对每行显示的记录数以及每页显示的记录数进行改变。

1. 设置每行包括的记录数

默认时每行 3 条记录。其代码如下。

```
<asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1"
            GroupItemCount="3">
```

其中，GroupItemCount="3"代表每行 3 条记录(3 列)。允许改变这个数值。

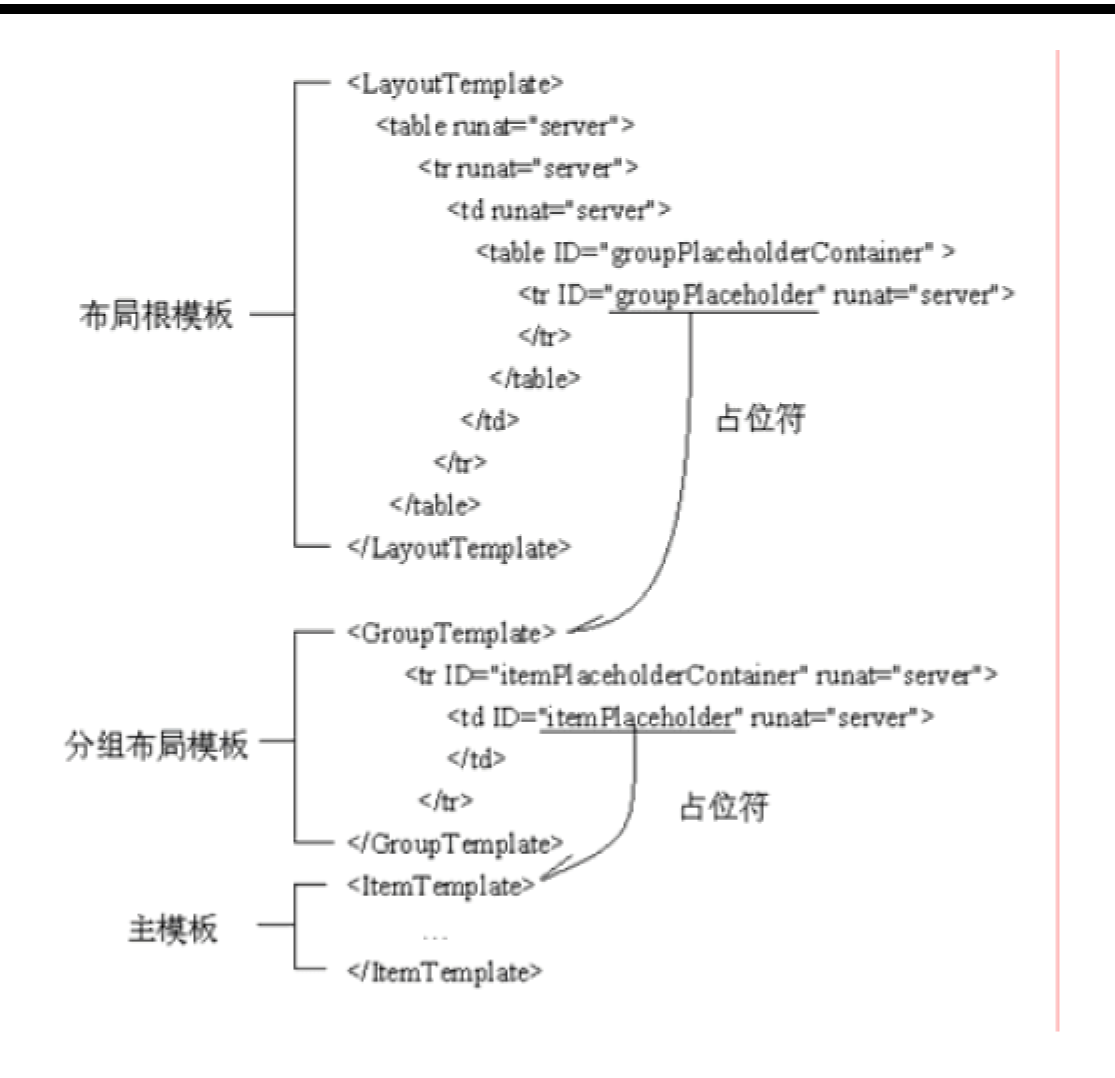


图 15.6 增加 GroupTemplate 模板后的代码

2. 设置每页包含的记录数

默认时每页 12 条记录。在 DataPager 控件中的代码如下。

```
<asp:DataPager ID="DataPager1" runat="server" PageSize="12">
```

如果将 PageSize="6"(即每页 6 条记录)时，显示的结果如图 15.7 所示。

注意：这些参数也可以在 ListView 控件的属性对话框中进行设置。

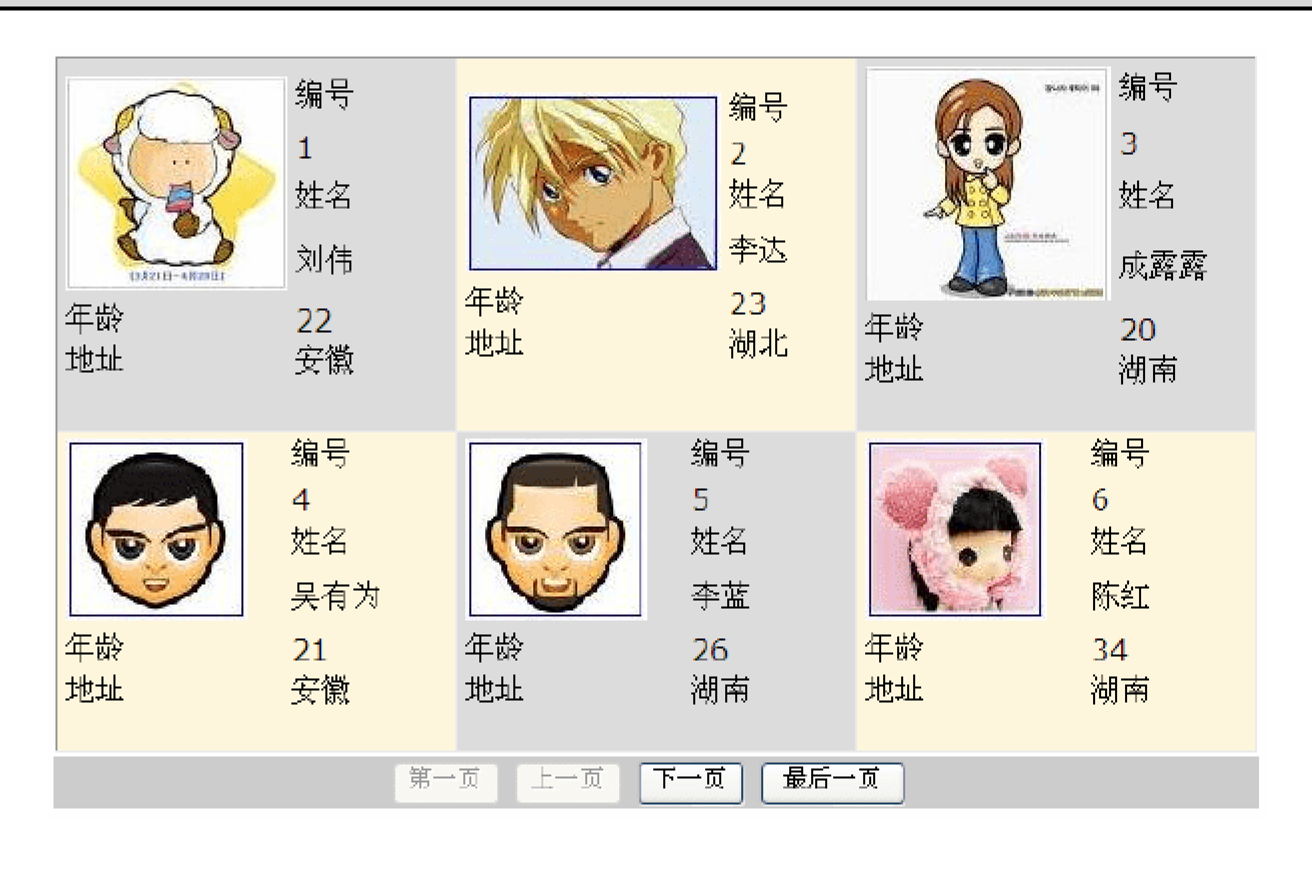


图 15.7 用平铺方式显示结果

15.5 小 结

ListView 控件是 ASP.NET 3.5 综合其他控件一些特点后推出来的一种新控件，新控件既能以表格方式显示数据，又能以平铺方式显示数据。用平铺方式显示数据最大的优点是

布局格式完全自定，特别适合于那些需要突出图像的界面。

ListView 控件中使用了大量的模板技术，其中只有 ItemTemplate 与 LayoutTemplate 两种模板是必需的，其他 8 种可以根据情况需要选择使用。

DataPager 是用于分页的控件，当我们在配置 ListView 控件时选中了【启用分页】项时，该控件就会自动加入到 ListView 控件的 LayoutTemplate 模板中。

15.6 习　　题

1. 填空题

(1) 在 ListView 控件中有很多模板，但是只有 ______________， ______________是必需的，其他都是可选的。

(2) 在配置 ListView 对话框中【启用编辑】、【启用插入】、【启用删除】选项不能使用，常常是因为___________控件的配置中没有使用________按钮。

(3) 当给 ListView 控件进行配置时，选中【启用分页】项时，系统就自动地增添了一个___________控件。

(4) 现在要在 ItemTemplate 模板中显示 age 字段(假定已经建立好与数据库的连接)的代码是

```
<asp:Label ID="ageLabel" runat="server" Text='____________________' />
```

2. 选择题

(1) 在 ListView 控件中 EditItemTemplate 是________，ItemTemplate 是________，AlternatingTemplate 是______，GroupTemplate 是______。

A. 编辑记录模板　　B. 交替记录模板

C. 分组布局模板　　D. 绑定数据的主模板

(2) 在 ListView 控件中选择【平铺】布局的最大好处是____________。

A. 设计简便　　B. 包含的数据容量大

C. 布局的格式自定　　D. 容易掌握

(3) 利用<%# Eval("age") %>方法能够____，利用<%# Bind("age") %>方法能够_____。

A. 编辑 age 字段　　B. A+D

C. 不能显示 age 字段　　D. 显示 age 字段

3. 判断题

(1) 在 ListView 控件中，ItemTemplate 是必须选用的模板。　(　)

(2) 在 ListView 控件中，EditItemTemplate 是必须选用的模板。　(　)

(3) 在 ListView 控件中，LayoutTemplate 是必须选用的模板。　(　)

(4) DataPager 控件的作用只用于分页。　(　)

4. 简答题

(1) 在 ListView 控件中用网格布局与平铺布局的主要区别是什么?

(2) 在平铺布局中下列语句中带下划线的字符串代表什么意思?

```
<asp: ImageID="…" runat="server" ImageUrl='<% #Eval("image") %>' />
```

5. 操作题

(1) 利用 ListView 控件用网格(表格)布局方式显示一个数据表(包括图片)。

(2) 利用 ListView 控件用平铺方式显示一个数据表。除图片必需以外，其他显示项自行确定。

第 16 章　存储过程与数据缓存

“存储过程”是数据库提供的功能，一些大型数据库，如 SQL Server、Oracle、DB2、Informix 等都提供了这种功能。它的特点是，允许将对数据库操作的各种 SQL 命令经过编译后直接存放到数据库端，以便提供服务。各个应用程序只需利用简单的调用语句，就像调用函数和过程一样即可调用存储过程，以完成对数据库的各项操作。这样，也就大大提高了代码的重用度，增强了程序的可靠性和运行效率。

除此之外，本章还要介绍数据缓存技术，它是提高程序运行效率的一项重要措施。本章中将介绍网页缓存和数据库缓存两个方面。本章中将要讲述的问题包括:

- 概述。
- 创建存储过程。
- 调用存储过程。
- 数据缓存。

16.1　概　　述

存储过程(Stored Procedure)是放置在数据库端的一组经过编译的、以 SQL 语句为基础的命令集。在 SQL Server 数据库的存储过程中使用的是 T-SQL，该语言既包括 SQL 语句，还允许包括一些过程语句。一个最简单的存储过程语句的格式如下：

```
CREATE PROCEDURE 存储过程名
(
  待定参数名 1 类型 1,
  待定参数名 2 类型 2,
  待定参数名 3 类型 3,
  …
)
AS
SQL 语句。语句中包括待定参数的赋值语句
```

下面是两个示例。

例 16.1　显示数据表 gyb 中的全部记录。

```
CREATE PROCEDURE gybSelectProcedure
AS
Select * From gyb
```

这是一个显示 gyb 数据表的存储过程。这个存储过程中不带任何待定参数。

例 16.2　更新数据表中的记录。

下面是一个带参数的存储过程，这个存储过程的作用是更新一个数据表。存储过程的名字是 UpdateCart，被更新的表名是 Cart。

```
CREATE PROCEDURE UpdateCart
(
 @OrderID uniqueidentifier,
 @OrderDate dateTime,
 @OrderNumber int
)
AS
Update Cart
SET OrderID = @OrderID,
    OrderDate = @OrderDate,
    OrderNumber = @OrderNumber
WHERE  OrderID = @OrderID
```

上述这个存储过程的作用是更新数据表的记录。其中“CREATE PROCEDURE…AS…”都是保留字。当创建完存储过程以后，再次调出来查看时，会发现前面的语句变成了 ALTER PROCEDURE。这代表现在是对语句的修改。

存储过程的语句中实际上包括两个不同的组成部分。

- 过程名、待定参数及其类型。如果存储过程中包括参数时，参数名及其类型都要放在小括号中，各参数之间用逗号分隔。
- AS 后面是存储过程命令的主体，包括可以执行的 T-SQL 语句。

存储过程是经过编译的、存放在数据库端的 SQL 命令集，应用程序只需调用存储过程的名字(需要时赋予相关的参数)，即可完成对数据库的操作，因而具有以下优点。

- 可以为多个应用程序共用，提高了代码的重用度。
- 运行可靠而且执行效率高。
- 能减少网络上的传输量。如果应用程序与数据库不在同一台机器上，而且使用的 SQL 语句比较复杂时，这个优点将更加明显。

16.2 创建存储过程

在 ASP.NET 3.5 中创建存储过程可以采用两种方式。不论哪种方式都需要先建立数据表。现在假定数据表的结构如表 16.1 所示。

表 16.1 数据表的字段设置

字 段 名	类 型	字 段 名	类 型	字 段 名	类 型
Bh(编号)	int 4 自动增值	XB(性别)	nvarchar 4	Phone(电话)	nvarchar 30
XM(姓名)	nvarchar 12	Age(年龄)	int 4		

16.2.1 在 SQL Server 2000 中创建存储过程

打开 Microsoft SQL Server 的企业管理器，并展开数据库名称的节点，单击【存储过程】节点。如果打开的是系统提供的样板库(如 Northwind、Pub 等)，将发现，系统已经提供了很多存储过程，双击其中之一就可以查看到该存储过程的代码。

为了创建新存储过程，右击【存储过程】图标，在弹出的快捷菜单中选择【新建存储过程】命令，将打开如图 16.1 所示的对话框。



图 16.1　在 SQL Server 数据库中创建存储过程

用存储过程的名称(如 InsertFromName)替换[OWNER][PROCEDURE NAME]。由于要使用参数查询，所以还必须在名称和 AS 关键字之间指定参数。代码如下。

```
CREATE PROCEDURE InsertFromName
(
    @Xm nvarchar(12),
    @XB nvarchar(4),
    @Age int
   )
AS
Insert INTO gybInsert
(Xm,XB,Age) Values (@Xm,@XB,@Age)
SELECT * FROM gyb
```

上面的语句包括两部分：第一部分用于插入记录(Insert)；第二部分用于显示数据表(Select)。当输完以上数据后可以单击【检查语法】按钮来检查语法的正确性。然后单击【确定】按钮以保存存储过程，如图 16.2 所示。

注意： 其中关键字(bh)应该自动产生增值，不应作为参数输入；字符串用 varchar 定义，而不能用 string。

16.2.2　直接在应用程序的环境中创建存储过程

可以在应用程序的环境中创建存储过程，这种方法更加直接。此时先打开数据库资源管理器，然后展开连接的数据库，再右击【存储过程】，如图 16.3 所示，在弹出的快捷菜单中选择【添加新存储过程】命令。以后的设置方法与前面所述相同。

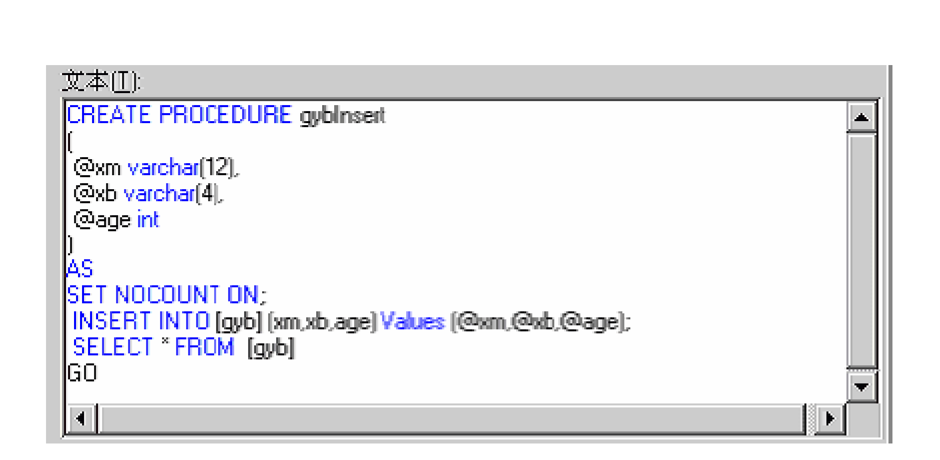




图 16.2　存储过程的语句示例

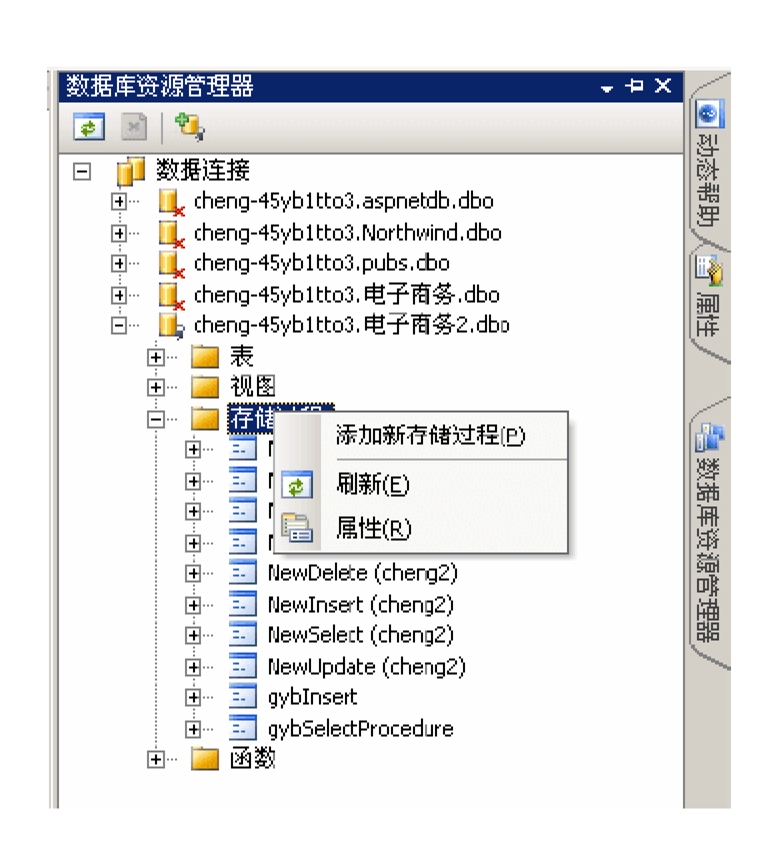


图 16.3　直接在应用程序中创建存储过程

16.3　调用存储过程

为了调用存储过程给数据表增添记录，在网页中放置几个 TextBox 控件以便输入参数，再放置一个 GridView 控件和一个按钮控件。界面设置如图 16.4 所示。

配置数据源控件以调用存储过程。步骤如下。

(1)　在配置数据源控件的过程中选用【指定自定义 SQL 语句或存储过程】，然后进行数据表编辑的设置。

(2)　选择【存储过程】，并在下拉列表中选择存储过程名。

(3)　确定存储过程中各待定参数的来源。具体情况如图 16.5 所示。

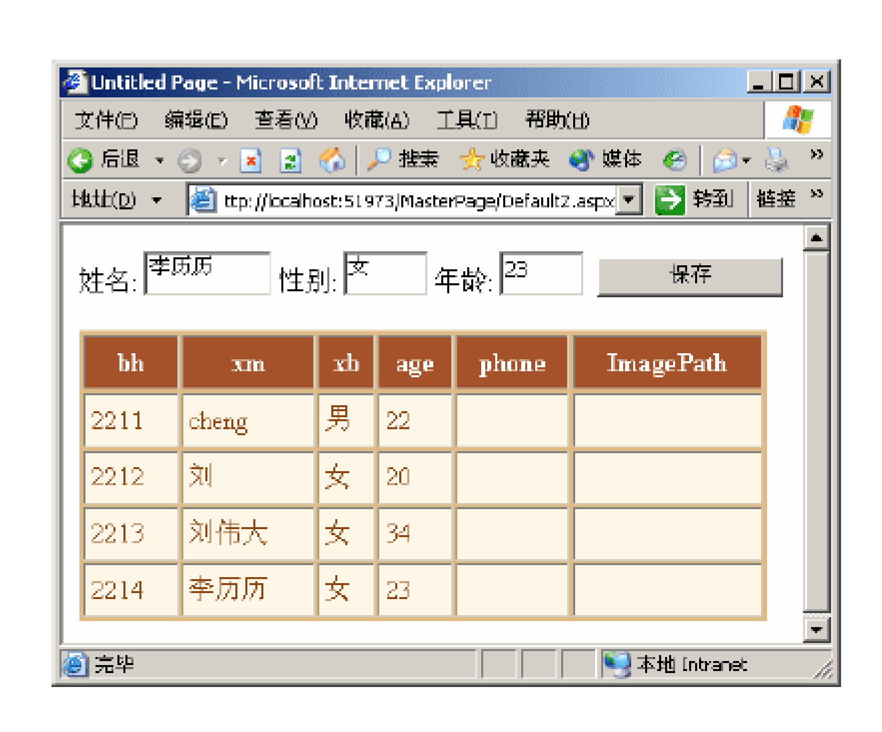


图 16.4　为调用存储过程的显示界面

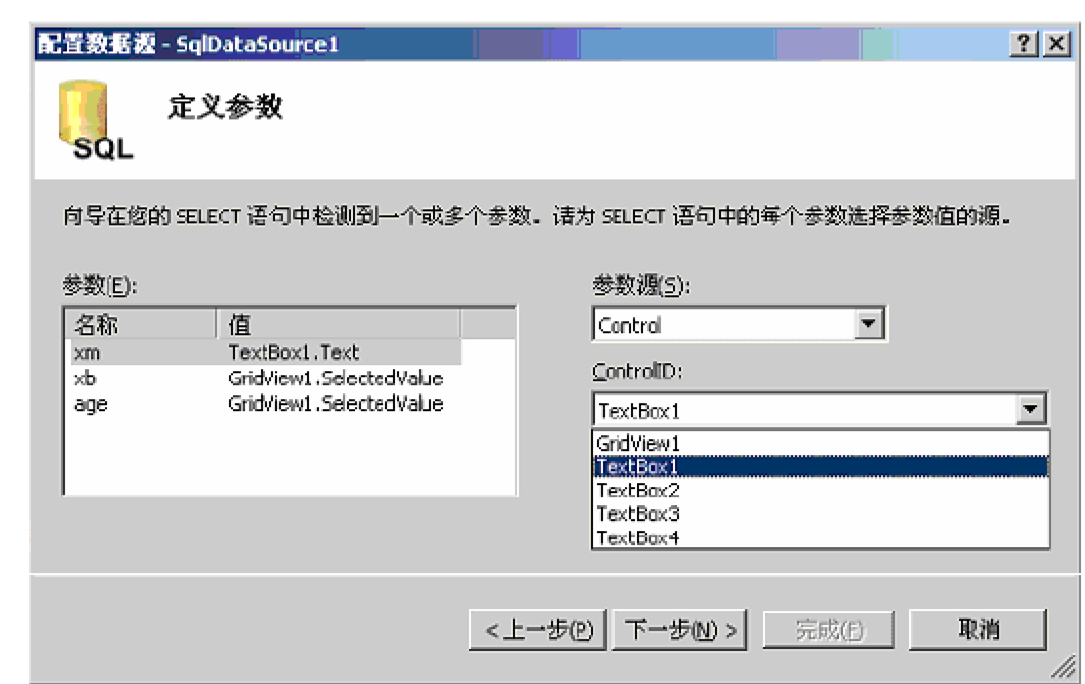


图 16.5　给存储过程确定参数来源

调用显示存储过程的方法与前面基本相同。相应的代码如下。

```
<%@ Page Language="C#" %>
<html>
<head runat="server">
  <title>GridView Bound to Stored Procedure</title>
</head>
```

```
<body>
  <form id="form1" runat="server">
    <asp:GridView ID="GridView1" DataSourceID="SqlDataSource1"
         AutoGenerateColumns="False"  runat="server">
      <Columns>
        <asp:BoundField DataField="TenMostExpensiveProducts"
             HeaderText="Product" />
        <asp:BoundField DataField="UnitPrice" DataFormatString="{0:c}"
             HeaderText="Price" />
      </Columns>
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
         SelectCommand="Ten Most Expensive Products"
         ConnectionString="<%$ ConnectionStrings:Northwind %>"
         SelectCommandType="StoredProcedure" />
  </form>
</body>
</html>
```

16.4　数 据 缓 存

数据缓存(Data Caching)就是将数据暂存于内存缓存区中的一种技术。当数据本身改变得不怎么频繁，而被访问的频率又比较高时，采用这种技术将大大提高数据访问的效率。

16.4.1　网页输出缓存

当网页的内容相对固定时，可以将整个网页缓存起来。因为对于动态网页来说，网页的访问大体上可以分为三个步骤。

(1) 客户请求。

(2) 动态生成网页并转化为 HTML 格式。

(3) 向浏览器发送显示。

如果网页首次被访问时将它缓存起来，后续客户再次请求时，就直接从缓存区中取出，发送显示，从而省略了“动态生成网页并转换为 HTML 格式”这个最费时的环节。

设置网页输出缓存(Output Caching)的方法很简单，下面通过一个简单的示例来说明。

(1) 在网站中增加一个网页，放入一个 Label 控件，假定控件的 id 名为 TimeMsg。

(2) 在网页的 Page_Load 事件中编写以下代码，以便在 TimeMsg 控件中显示打开网页的时间。

```
protected void Page_Load(object sender, EventArgs e)
  {
    TimeMsg.Text = "打开网页的时间是:"+DateTime.Now.ToString();
  }
```

(3) 在*.aspx 网页的代码中增加设置缓存的指令。

```
<%@ OutputCache  Duration="600"  VaryByParam=none %>
```

语句“<%@...%>”是网页配置的指令，在这里用来给网页指定缓存参数。其中，Duration="600"(注意：时间两端要加引号)代表缓存持续时间为 600 秒，VaryByParam 属性用来指定特定版本的网页输出。在<%@OutputCache...%>配置指令中一定要加入 VaryByParam 属性。即使不使用这个版本属性，也要将它加入，但将其值设为 none。

为了演示缓存过程，打开网页后，按前面的要求设置<%@OutputCache...%>指令，并用控件显示当前的时间。然后多次“刷新”网页(代表对网页的多次访问)。将看到在缓存的持续时间内显示的时间不会改变，这说明显示的是从缓存区中取出来的副本。当持续时间到达时显示的时间才会变化，同时又开始新一轮的缓存。

如果网页中包含有对网页请求的 URL 查询字符串，因为查询结果是动态生成的，再按照上述方法设置缓存指令会有问题。此时应采用其他可选择的办法。即将<%@ OutCache>页面指令中的 VaryByParam 属性设为“*”，以表明页面中使用了查询语句，应该根据不同的查询参数来缓存页面的多个不同副本。

还可以进一步将 VaryByParam 属性直接设为“查询参数名”。例如：

```
<%@ OutputCache  Duration="600"  VaryByParam="ProductID" %>
```

这种情况下 ASP.NET 将检查查询字符串，以查找 ProductID 参数，并对不同的 ProductID 单独缓存其不同的副本。

网页缓存以后，不论访问的客户来自世界何处，都直接从缓存区中提取出副本发送出去。

16.4.2 利用数据源控件缓存数据库

通常情况下，大量数据是保存在数据库中的，而应用程序访问数据库是一项非常费时的操作。因为访问数据库时，先要连接并打开硬盘中的数据库，执行查询或编辑数据的命令，取出数据后还要关闭数据库等。这个过程需要系统付出不小的开销。如果经常重复这些操作，必然会大大增加系统的负担，降低系统的运行效率。

如果先将数据库中的数据缓存到缓存区中，当应用程序需要这些数据时，直接从缓存区中提取，就可以显著地减少系统开销，提高系统的运行效率。

在 ASP.NET 中可以通过数据源控件(如 SqlDataSource)的属性来设置数据表的缓存参数。缓存参数包括以下几项，情况如图 16.6 所示。

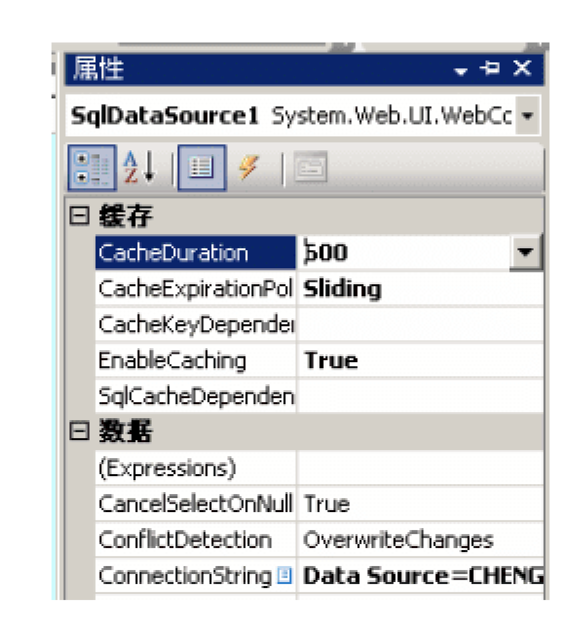


图 16.6 在数据源控件的属性中设置缓存属性

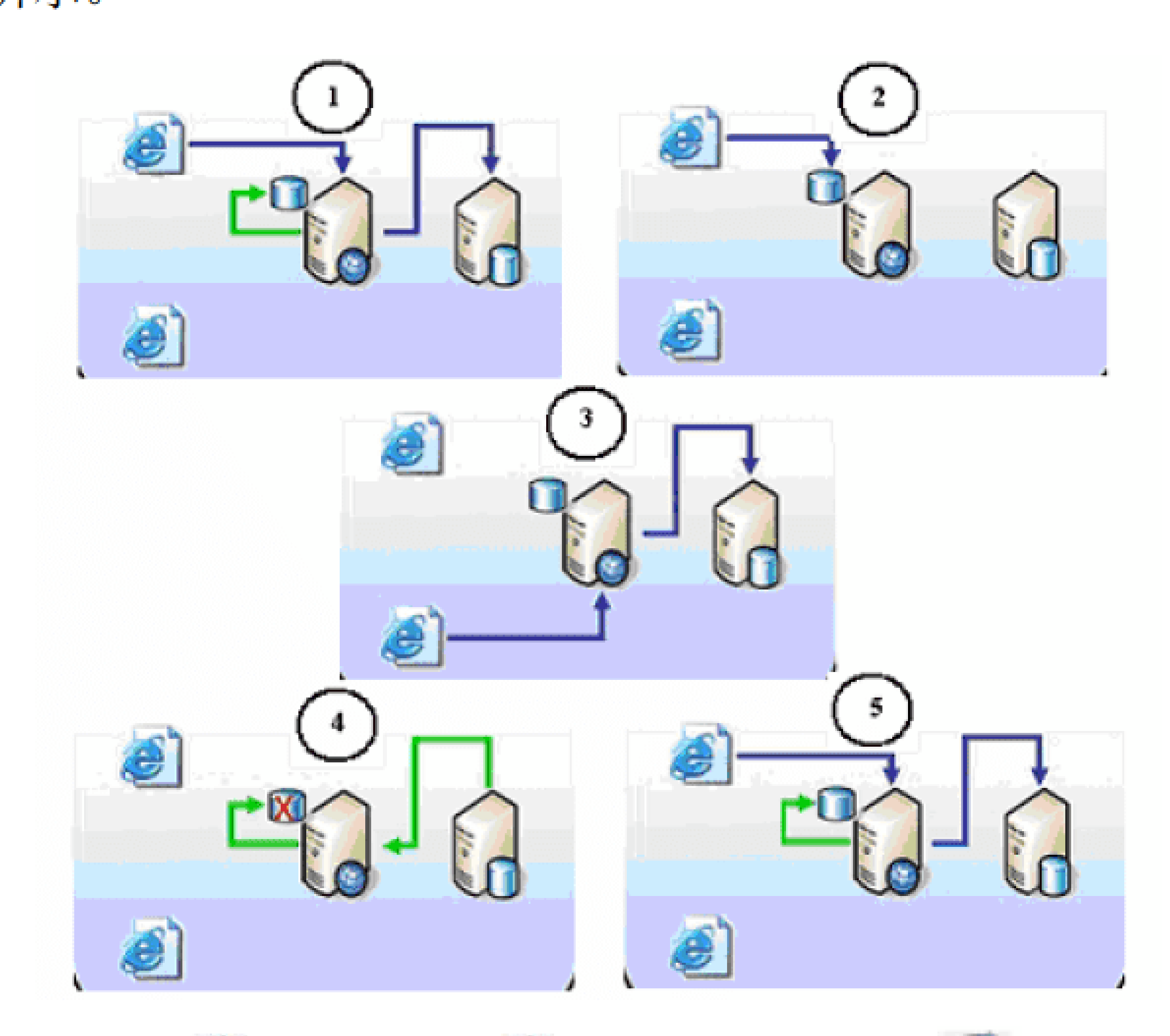
- EnableCahing：默认时为 false，即不使用数据缓存，将该属性改为 true 时即可启动数据缓存。
- CacheDuration：代表缓存的持续时间。默认时为 Infinite(无限)。本例中设置成 600 秒。
- CacheExpirationPolicy：缓存策略。包括两种设置：Absolute 和 Sliding。设置成 Absolute 时，时限一到缓存区失效。对于需要定期更新的信息来说，选择这种策

略比较合适；当设置为 Sliding 时，时限到时立即刷新缓存区中的数据，并继续缓存更新后的数据。

- CacheKeyDependency：用于创建与其他文件或缓存关键字相依赖的关系。
- SqlCacheDependency：用于创建与 Microsoft SQL Server 数据库的依赖关系。

简单的缓存只需设置前三项，即将 EnableCahing 属性设为 true；将 CacheDuration 指定缓存时间；将 CacheExpirationPolicy 指定缓存策略。

利用数据源控件进行缓存有一个很大的好处，就是如果该数据源控件既担负查询(Select)又担负编辑(Update)任务时，一旦在缓存持续时期内数据表的数据改变了，系统会自动将原来缓冲区的数据作废，并从数据表中重载缓存数据，以便使两种数据保持一致。其过程如图 16.7 所示。



代表客户；代表服务器；代表数据库管理器；代表内存空间

图 16.7　SQL Server 2008 数据库数据缓存的原理

图 16.7 表示用数据源控件缓存数据表的过程。这个过程是：

① 当某客户访问数据库时，取出数据的同时将该数据缓存在内存中。

② 当需要再次访问数据库时直接从内存缓冲区中读取数据。

③ 现在出现另一客户来修改数据库的情况。

④ 系统将内存中原来缓存的数据变为无效。

⑤ 自动从数据库中读取数据缓存到内存中。

利用数据源控件缓存数据表也存在问题，即当内存容量小于缓存的需要时，缓存区将会被自动删除，因而不能保证在整个缓存时间内持续缓存。

如果在网站中设有多处缓存区，缓存了一些相互关联的数据，数据源一旦有变化，多处缓存都应作废，否则将可能使用过时的数据。这些缓存区之间存在着一定的依赖关系，需要设置一个通知功能。CacheKeyDependency 用来创建文件之间的依赖关系，SqlCacheDependency 用来创建数据库之间的依赖关系。为了建立它们之间的依赖关系需要

有较多的设置，由于篇幅所限，这里不再讲述。

16.5 小　结

存储过程是存放在数据库端经过编译的，由多条 SQL 语句(还可包括一些过程语句)组成的程序。一些大型数据库都提供了这一功能。可以直接在数据库中，也可以在网页中创建存储过程。调用存储过程的语句类似于调用其他方法的语句，关键在于正确地给各待定参数赋值。

数据缓存是将网页或数据库中的数据暂存于内存的一种方法。数据缓存可以大大提高系统运行的效率。

在 ASPX 网页中，通过<%@ OutputCache Duration=n VaryByParam=none %>配置网页的缓存。通过数据源控件(如 SqlDataSource)来缓存数据库，简单情况下只需要对数据源的三个属性进行设置即可。利用数据源控件来缓存数据库的最大好处是，它可以自动更新过时的缓存数据。

16.6 习　题

1. 填空题

(1) 存储过程是用各种 SQL 命令编写并经过编译后直接存放到____________端的程序。

(2) 在 SQL Server 数据库中存储过程使用的是____________语言。

(3) 一个简单的存储过程的代码包括两部分:

```
CREATE PROCEDURE 存储过程名
(
  // 第一部分
)
AS
  // 第二部分
```

其中第一部分是______________，第二部分是___________________。

(4) 下面是一段网页缓存的指令。

```
<%@ OutputCache  Duration="40"  VaryByParam=none %>
```

其中:

Duration="40"代表____________________。

VaryByParam=none 代表________________。

2. 选择题

(1) 在 SplDataSource 数据源控件中，若将数据库缓存的 CacheExpirationPolicy 属性设置为 Absolute 时，缓存时限一到则__________。

A. 自动延长缓存时间　　B. 刷新缓存区并继续缓存

C. 延长一倍缓存时间　　D. 缓存区失效

(2) 在 SqlDataSource 数据源控件中，若将数据库缓存的 CacheExpirationPolicy 属性设置为 Sliding 时，缓存时限一到则__________。

A. 自动延长缓存时间　　B. 刷新缓存区并继续缓存

C. 延长一倍缓存时间　　D. 缓存区失效

3. 判断题

(1) 所有的数据库都可以使用自己定义的存储过程。　　(　　)

(2) 在 T-SQL 中既包括 SQL 语句还可以包括过程语句。　　(　　)

4. 简答题

(1) 使用存储过程的好处是什么？

(2) 什么情况下可以进行网页缓存？网页缓存带来什么好处？

(3) 试述利用数据源控件(SqlDataSource)缓存数据表时设置缓存策略以及缓存持续时间的方法。

(4) 利用数据源控件(SqlDataSource)缓存数据表的最大好处是什么？

5. 操作题

(1) 实际编写和调用存储过程，以便向数据表中增添记录。

(2) 通过设置完成对某网页的输出缓存工作。

(3) 进行数据表缓存。

第 17 章　创建三层架构

前面讲述的网站都是属于两层架构(C/S 结构)。就是说，网站中的显示层(包含处理逻辑的代码)直接与服务层连接。随着网站功能的增强，网站结构也变得复杂起来，此时就需要对系统作进一步分类、封装和抽象。三层架构(3-tier)的出现适应了复杂网站的需要，目前已经变得越来越普遍。所谓三层架构就是在客户的显示层与服务器层中间增加一个中间层。在中间层中放置网站共用的逻辑处理代码，在电子商务中通常是一些商务规则和商务逻辑处理代码。

ASP.NET 2.0 对三层架构提供了有力的支持，ASP.NET 3.5 在此基础上又作了很多改进。下面将重点讲述 ASP.NET 3.5 的三层架构的新特点。本章将要讲述的问题包括：

- 从两层架构发展成三层架构。
- ASP.NET 3.5 中间层的特点。
- 创建中间层的步骤。
- 在网页中调用中间层中的对象。
- 三层架构的应用示例。

17.1　从两层架构发展成三层架构

传统的两层架构就是客户机/服务器模式。在这种模式中，客户向服务器发出请求，服务器处理这些请求，处理完成以后再返回给客户端。此时显示代码和逻辑处理代码都集中于前台的网页之中。如果系统的功能比较简单时，非常适合采用两层架构。

当系统的功能比较复杂，或者对网站有些特殊要求时，最好改用三层架构来取代两层架构。三层架构的核心思想是，将整个应用划分成三层：表示层—业务层—数据访问层(含数据库)。也就是在客户机与服务器之间增加一个中间层(有时又称为业务组件层)，用来放置处理业务的逻辑代码。两层架构与三层架构的示意如图 17.1 与图 17.2 所示。

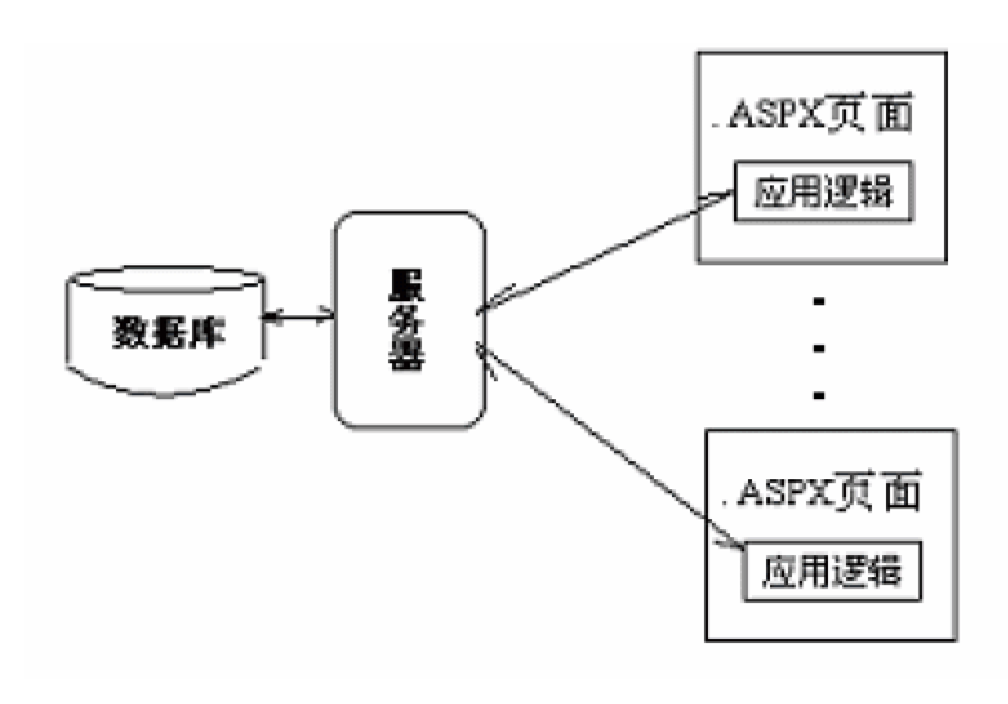


图 17.1　两层架构示意

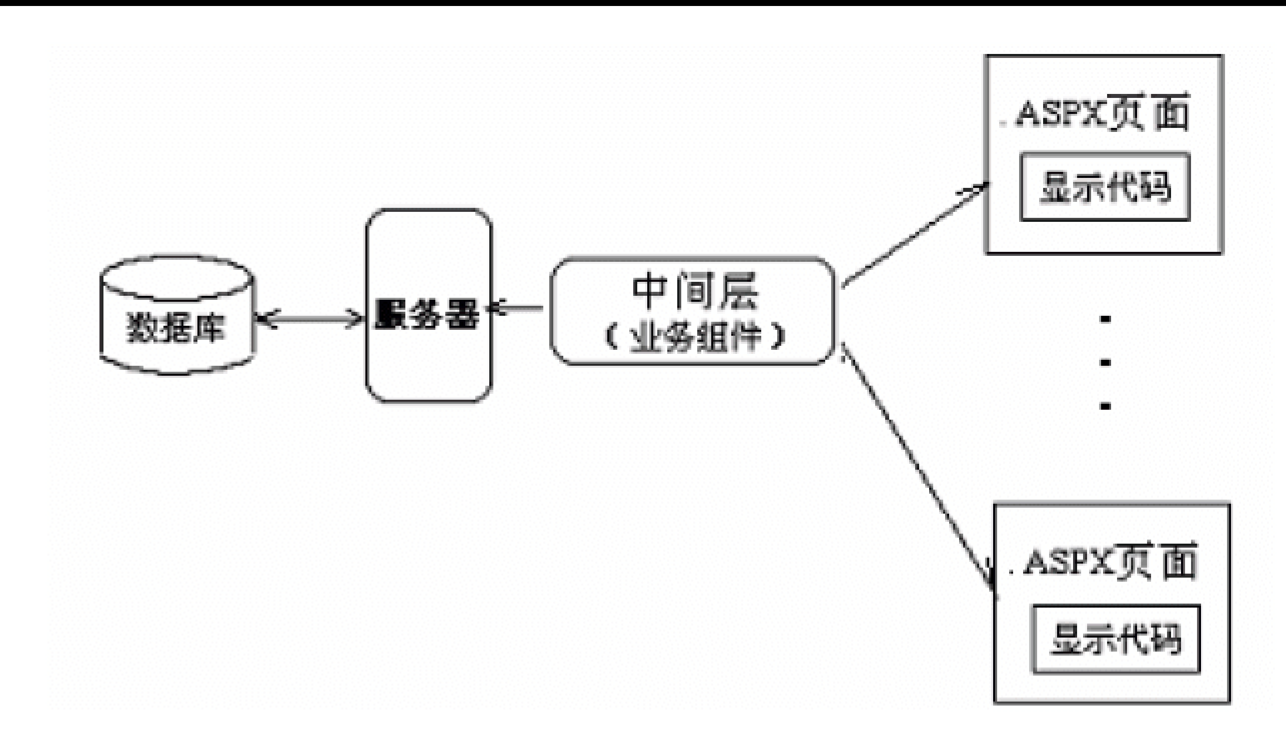


图 17.2　三层架构示意

在三层架构中，客户端网页是系统的前台，负责客户界面的显示，其他非显示(非 UI)的逻辑处理部分(包括业务规则或商业逻辑)都集中放在中间层中。后台则负责数据的存储和管理。这样的分工不仅思路清晰，代码重用度高，而且一旦商务逻辑或业务规则需要改变时，只需对中间层进行修改而不需要分别对各个网页进行修改。这样做有利于系统的维护和扩展，还可防止各窗体中出现不一致的现象。

这里所谓的“三层”是指逻辑上的划分。有些系统的商业逻辑比较复杂，需要在物理上再划分成多层。但这些物理上的多层，逻辑上都可看成中间层。这就好比一部戏剧的演出，舞台上的演员属于第一层，他(她)们面对的是广大观众；而管弦乐队、舞台管理人员和导演属于第二层，他们只和舞台上的演员打交道，观众看不到他们；剧本的作者、布景师等属于第三层，观众不能看到他们而只能感受到他们的作品。

17.2　ASP.NET 3.5 中间层的特点

中间层实质上是一些不含显示界面(UI)的“类”的集合。从理论上来讲，任何类都可以放在中间层中，这些类的作用是对众多的方法和属性作进一步的封装和抽象。设计者可以从头开始设计中间层的类，也可以先在类库中选择一个功能相近的类，然后在这个基础上进行继承或扩展。后一种方式不仅能简化设计过程，还能增加新类的可靠性，符合面向对象的设计思想。

ASP.NET 3.5 中提供的对象数据源控件(ObjectDataSource)、数据集(DataSet)和数据表适配器(TableAdapter)为通过中间层访问数据库提供了强有力的支持。将这些控件放在中间层，供各个网页调用，将提高代码的重用度并保证各网页中数据的一致性。

这几个控件的关系如图 17.3 所示。该图包括 4 个组成部分，它们的作用分述如下。

- 数据集：数据集是内存中的数据库，可以在断开与数据源连接的条件下继续工作。数据集从数据库中获得数据以后，还可以增加一些字段以填充临时数据。
- 数据表适配器：数据表适配器是数据集与数据源的接口，用来连接数据库，执行检索数据表并向数据集填入数据、取出数据和更新数据等操作。数据表适配器的Fill()方法用来向数据集填入数据。一个数据表适配器可以执行多条检索语句，这些检索语句是对同一个数据表进行的检索，数据表的字段虽然相同，但是由于检

索条件(WHERE)不同，获得的数据也不相同。适配器采用 Fill()、FillBy()、FillBy1()等不同的方法名来区分不同的检索结果。与此相对应，取出数据时调用的方法也分别采用 GetData()、GetDataBy()、GetDataBy1()等不同的名字。当数据表适配器更新数据时调用 UpDate()方法。数据表适配器还可以根据需要定义各种不同的方法，提供给“类”调用，从而大大增强了使用的灵活性。

- 自定义类：自定义类是将各个数据集、数据集中的数据表以及数据表适配器封装到一起，以便于调用和处理。

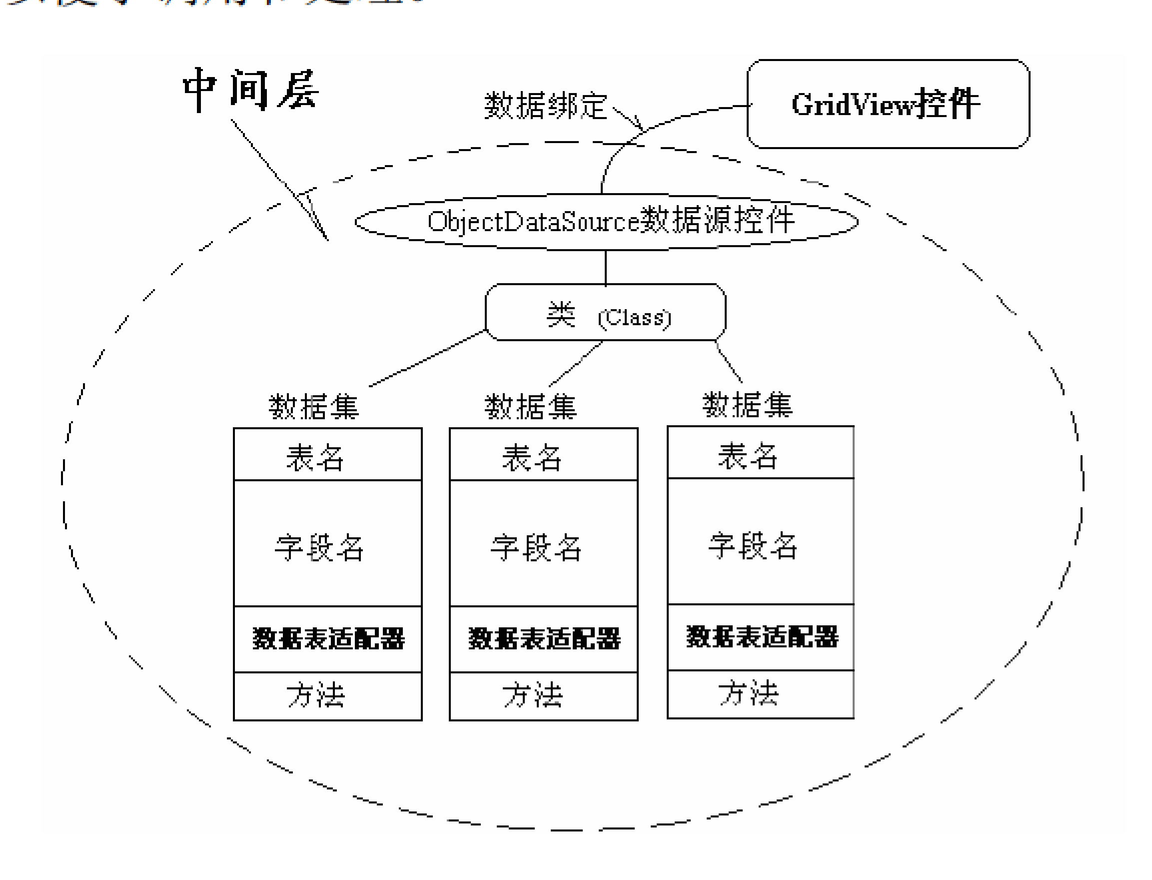


图 17.3 连接数据库的中间层

- ObjectDataSource 控件：它的一端连接自定义类，另一端与显示控件(这里用的是 GridView)进行数据绑定。除此之外，ObjectDataSource 控件还可以在以下几方面发挥作用。
  - ◆ 给数据分页和排序。
  - ◆ 缓存数据。
  - ◆ 防止数据访问中的冲突。

17.3 创建中间层的步骤

下面通过一个简单的示例来说明创建中间层的方法。创建的过程分为两步。

(1) 创建数据集与数据表适配器。

① 在网站中增加“数据集(DataSet)”，右击网站名，在弹出的菜单中选择【添加新项】命令。在弹出的对话框中选择【数据集】，然后给它取名。假定给它命名 ProductsDataSet.xsd，弹出的界面如图 17.4 所示。

② 单击【添加】按钮后，当出现“是否放置在 App_Code 目录中”的提示时，单击【是】按钮。

现在一个数据集已经被增加到网站中，同时还增加了一个“数据表适配器(TableAdapter)”。下面需要进一步给数据表适配器进行配置。给数据表适配器进行配置的方法与给 SqlDataSource 数据源控件的配置方法基本相同。

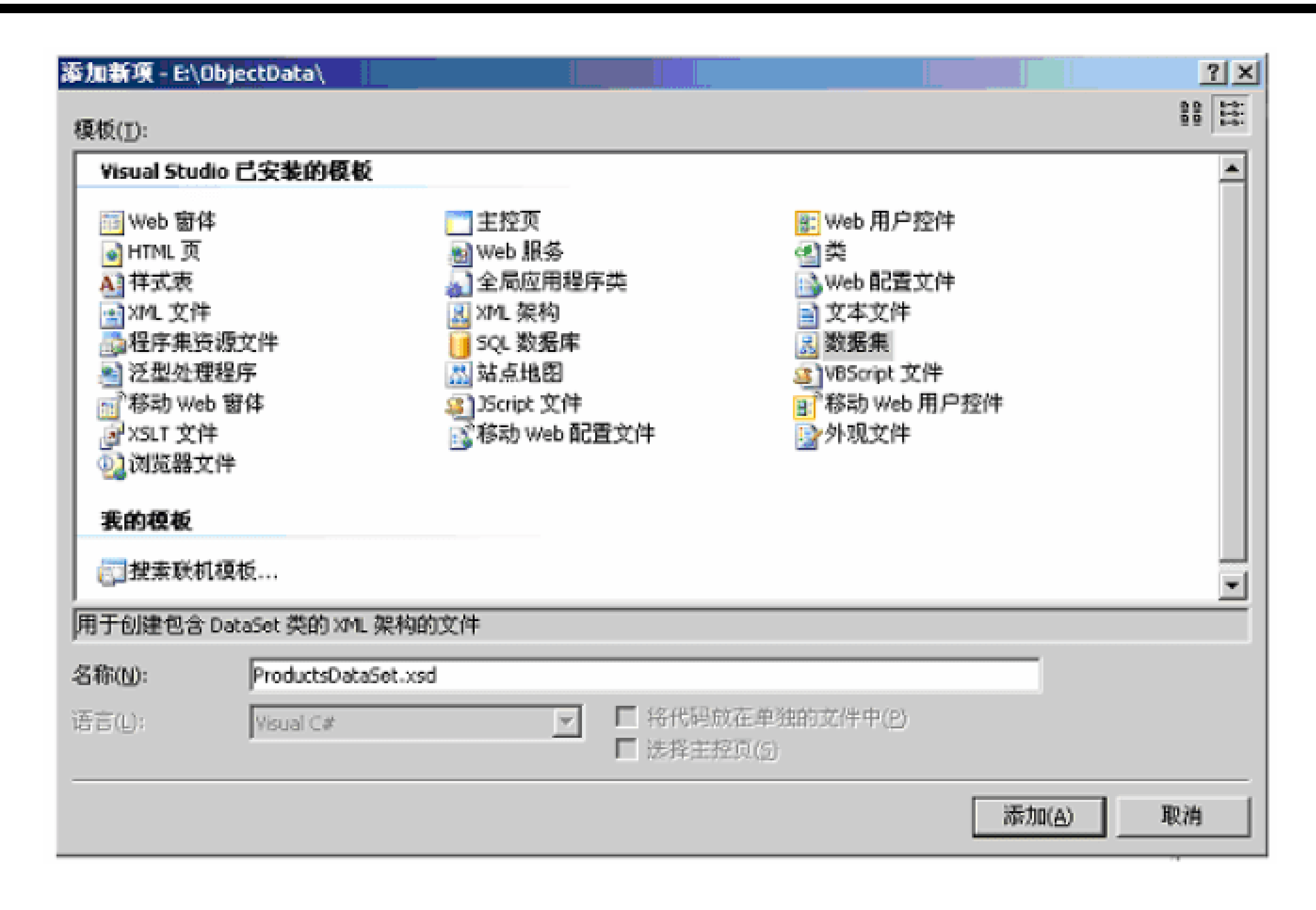


图 17.4　添加数据集

③　在数据表适配器向导的帮助下，先后两次单击【下一步】按钮。在【选择命令类型】的对话框中，选择【使用 SQL 语句】并选择 Products 数据表，选择字段以后将自动生成以下语句。

```
SELECT ProductsID, ProductName, CategoryID, QuantityPerUnit, UnitPrice
FROM Products WHERE (CategoryID =  @CategoryID)
```

④　单击【完成】按钮，并保存前面的设置，将看见如图 17.5 所示的界面。

⑤　在这个基础上可以进一步调整和扩展数据集与数据表适配器的功能。方法是右击数据集的标题栏，将弹出如图 17.6 所示的菜单。选择【添加】|【列】命令，可以在数据集中增加字段，以便放入一些临时数据；选择【添加】| Query 命令，可以为数据表适配器增加检索语句；选择【添加】| Relation 命令可以增添数据表之间的关系。

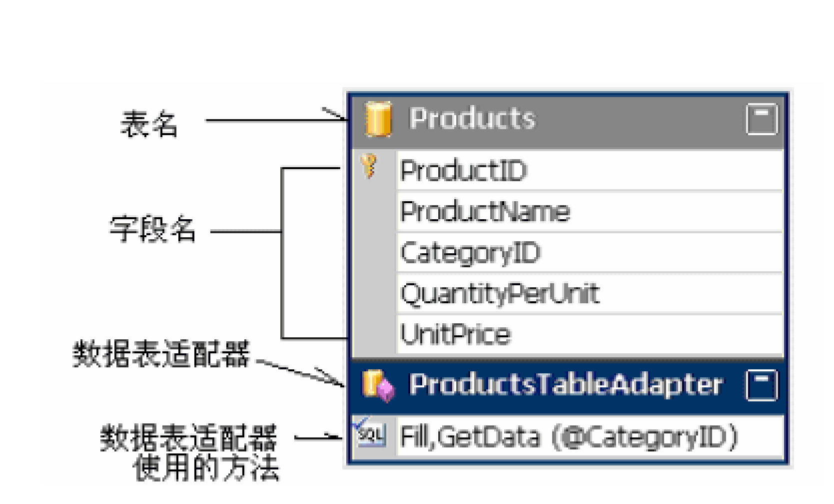


图 17.5　数据集及数据表适配器

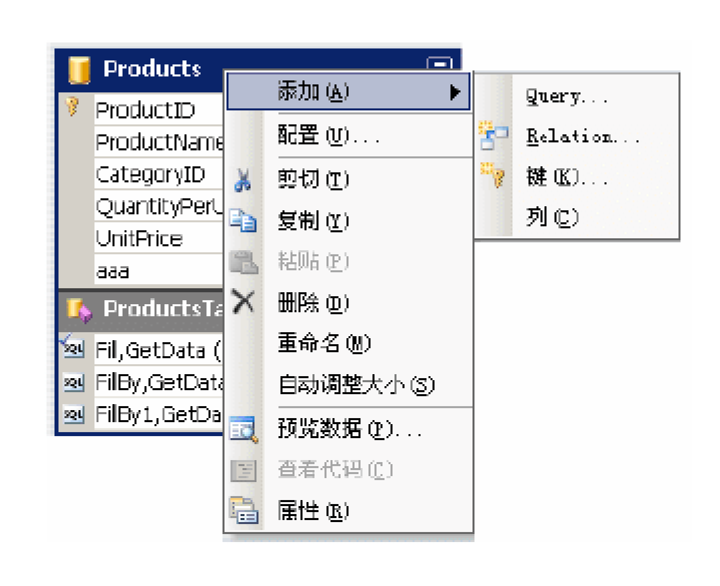


图 17.6　扩展数据集和数据表适配器功能

(2)　创建自定义类。

右击网站中的 App_Code 目录，选择【添加新项】命令。在弹出的对话框中选择【类】，并且给类命名。假定将该类命名为 DataAccess.cs，单击【添加】按钮。

```
public class DataAccess
{
public DataTable getProducts(int Cate)
{
```

```
ProductsDataSetTableAdapters.ProductsTableAdapter Ada = new
                    ProductsDataSetTableAdapters.ProductsTableAdapter();
                                //生成数据适配器对象 Ada
ProductsDataSet.ProductsDataTable DD = new
                    ProductsDataSet.ProductsDataTable();
                                //生成数据集对象
Ada.Fill(DD,Cate);
                                //将过滤的数据填入数据集中
return DD;                      //返回数据表
}
}
```

这段代码的作用是，先分别生成数据表适配器的对象(Ada)和数据集的对象(DD)，然后将适配器中检索的数据填入数据集中。其中 ProductsDataSetTableAdapters 是命名空间。这个命名空间是一个组合字，前面的 ProductsDataSet 是数据集的文件名，后面的 TableAdapters 代表数据表适配器。

17.4 在网页中调用中间层对象

在网页中可以用两种方式来调用中间层。

- 直接调用中间层定义的类对象。
- 通过 ObjectDataSource 数据源控件间接调用中间层。

下面分别讲述这两种方式。

17.4.1 直接调用中间层对象

具体操作步骤如下。

(1) 在网页中设置下拉列表框(DropDownlist)以及 GridView 控件，如图 17.7 所示。

(2) 为下拉列表框的 Items 属性设置参数，如图 17.8 所示。

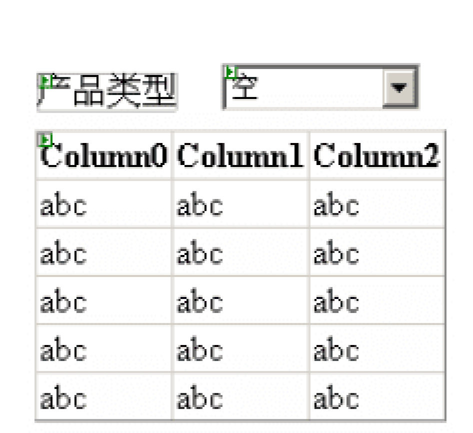


图 17.7 直接生成类对象的界面

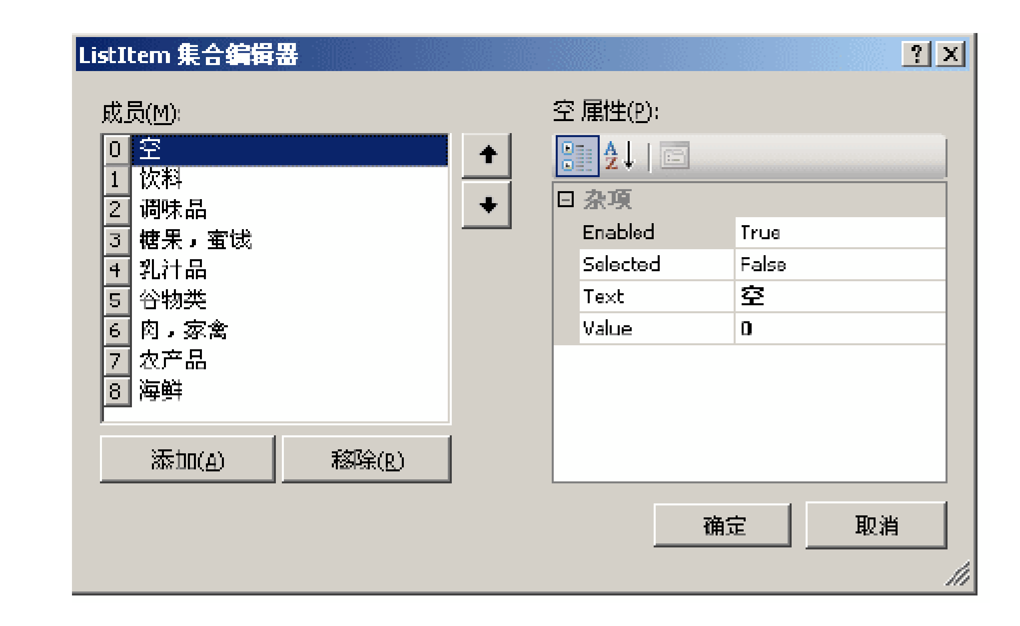


图 17.8 为下拉列表框设置 Items 属性

(3) 在网页的 Page_Load 事件中编写代码如下。

```
public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
```

```
    {
      DataAccess DA = new DataAccess();  //DataAccess 是前面定义的类名
      GridView1.DataSource = DA.getProducts(int.Parse(DropDownList1.SelectedValue));
      GridView1.DataBind();
    }
}
```

这段代码的含义是先生成自定义类的对象(DA)，再通过该对象中的方法(getProducts())取出表中的数据，作为 GridView 控件的数据源。

17.4.2 通过 ObjectDataSource 数据源控件调用中间层

具体操作步骤如下。

(1) 在网页界面中增加 ObjectDataSource 数据源控件。

(2) 配置数据源时先取消选中【只显示数据组件】复选框，然后在【选择业务对象】下拉列表框中选择新定义的类(DataAccess)，如图 17.9 所示。

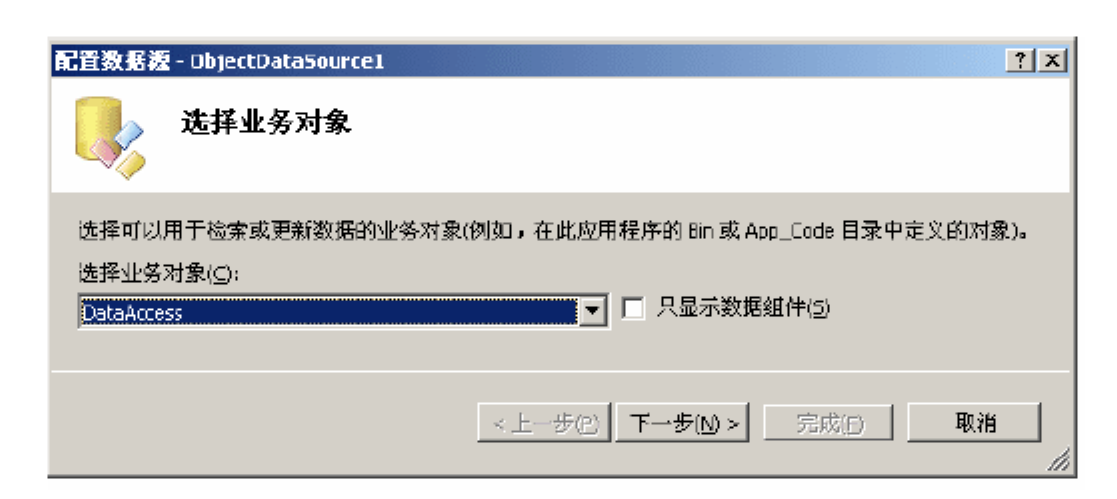


图 17.9 在对象数据源中选择业务对象

(3) 单击【下一步】按钮后，在【定义数据方法】界面中，打开 SELECT 选项卡，并在【选择方法】下拉列表框中选择类中定义的方法，如图 17.10 所示。

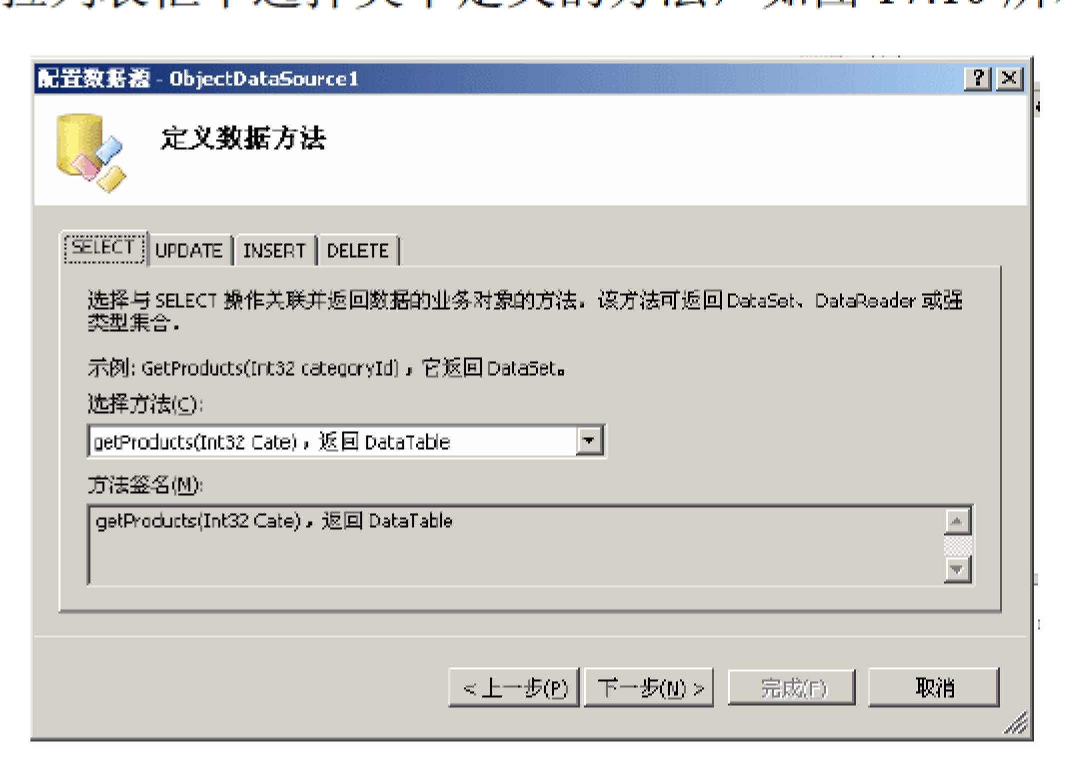


图 17.10 定义数据方法

(4) 单击【下一步】按钮后，在【定义参数】界面中设置参数源，如图 17.11 所示。到此为止，对 ObjectDataSource 控件的配置已经完成。此时该控件的 TypeName 属性中已经载明了新定义类的名字(此处为 DataAccess)。

(5) 将 GridView 控件的 DataSourceID 属性指向 ObjectDataSource1 数据源，即可运行

程序。



图 17.11　确定参数来源

注意：使用第二种方式时，第一种方式中对 GridView1 的 Page_Load 事件的设置应该取消。

采用第一种方式时，直接连接“类”虽然比较简单，但功能不强；采用第二种方式时可以利用 ObjectDataSource 控件提供的功能，增强控制能力并自动解决一些数据访问中可能出现的矛盾，因而是一种比较好的方式。

17.5　三层架构的应用示例

为了使问题简明易懂，这里只集中讲述在三层架构中对某个局部问题的解决方案。即在商品的交易过程中，用从客户预付款中扣除的方式来完成付款的工作。为此，必须先将客户的预付款保存在客户表中。客户表(Customers)如图 17.12 所示。



CustomerID	CompanyName	ContactName	Address	Phone	Premoney
3300	河北宏泰公司	刘福贵	秦皇岛路径路...	0335-64536783	8700.0000
3301	上海六福公司	李艳簿	上海常得路220号	021-88778899	6500.0000
3302	长沙实用商贸...	何时为	长沙51路235号	0731-77665544	5780.0000
3303	北京前门商店	吴作为	北京前门饭店6...	010-77885544	7690.0000

图 17.12　客户表示例

当客户选择商品后，提交订单时需先执行以下过程，然后才能存储订单。

(1)　取出客户的预付款。

(2)　从预付款中减去本次交易应交的付款数。

(3)　将剩余的款项存回客户的数据表中。

如果在第(2)步中出现了负数，此次交易作废。

这是一种从数据表中取出数据→处理该数据→再将处理完的数据存回数据表的过程，这种过程在程序中可能会经常遇到。利用三层架构来处理很容易实现。处理的步骤如下。

(1)　为访问数据库创建存储过程。

应该创建足够的存储过程，以便需要时调用。这里只讲述其中的两个：一个用于读取预付款；另一个用于修改预付款。后者的作用就相当于将处理后的余额送回数据表。两个存储过程的定义如下。

- 读取预付款(GetPremoney)的存储过程：

```
ALTER PROCEDURE dbo.GetPremoney
    (
    @CustomerID int
    )
AS
  Select Premoney From Customers Where CustomerID = @CustomerID
```

- 修改预付款(UpdatePremoney)的存储过程：

```
ALTER PROCEDURE dbo.UpdatePremoney
    (
    @CustomerID nvarchar(20),
    @Premoney Decimal
    )
AS
    Update Customers SET Premoney = @Premoney WHERE CustomerID =
@CustomerID
    Select * From Customers
    RETURN
```

说明： 在数据表中预付款(Premoney)的类型是 money，对应于存储过程中的类型是 Decimal。

(2)　创建数据集及数据表适配器。

右击网站名，在弹出的菜单中选择【增加新项】命令，然后将数据集(DataSet1.xsd)增加进来(注意，数据集必须放在 App_Code 的目录下)，并配置数据集以连接 Customers 数据表。

然后右击【数据集】中的【数据表适配器】，在弹出的菜单中选择【添加查询】命令，将弹出如图 17.13 所示的对话框。

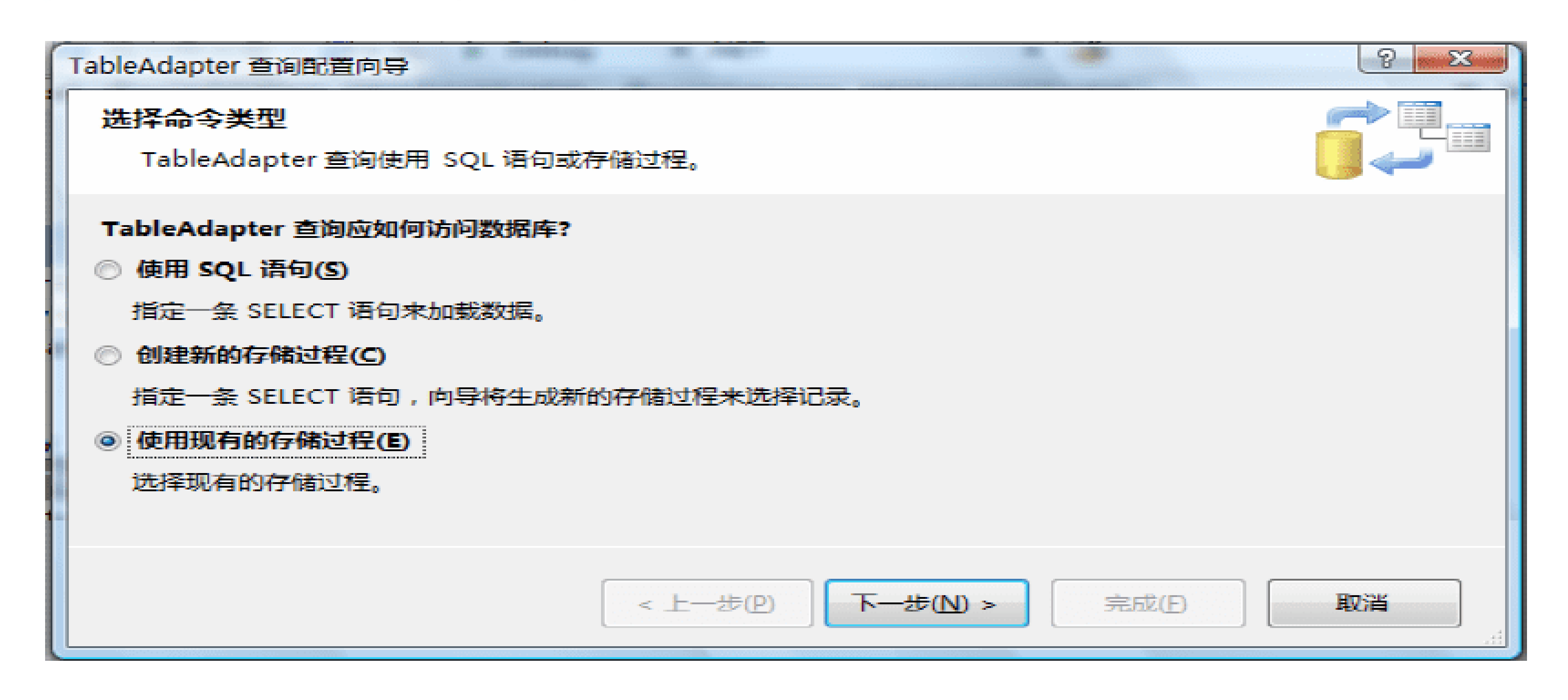


图 17.13　数据表适配器配置向导(1)

选择其中的【使用现有的存储过程】项，单击【下一步】按钮。在下拉列表框中选择

GetPremoney 存储过程，如图 17.14 所示。

图 17.14　数据表适配器配置向导(2)

对话框表明对于该存储过程来说，需要输入的参数是@CustomerID，输出的结果是 Premoney。

单击【下一步】按钮，弹出如图 17.15 所示的对话框。

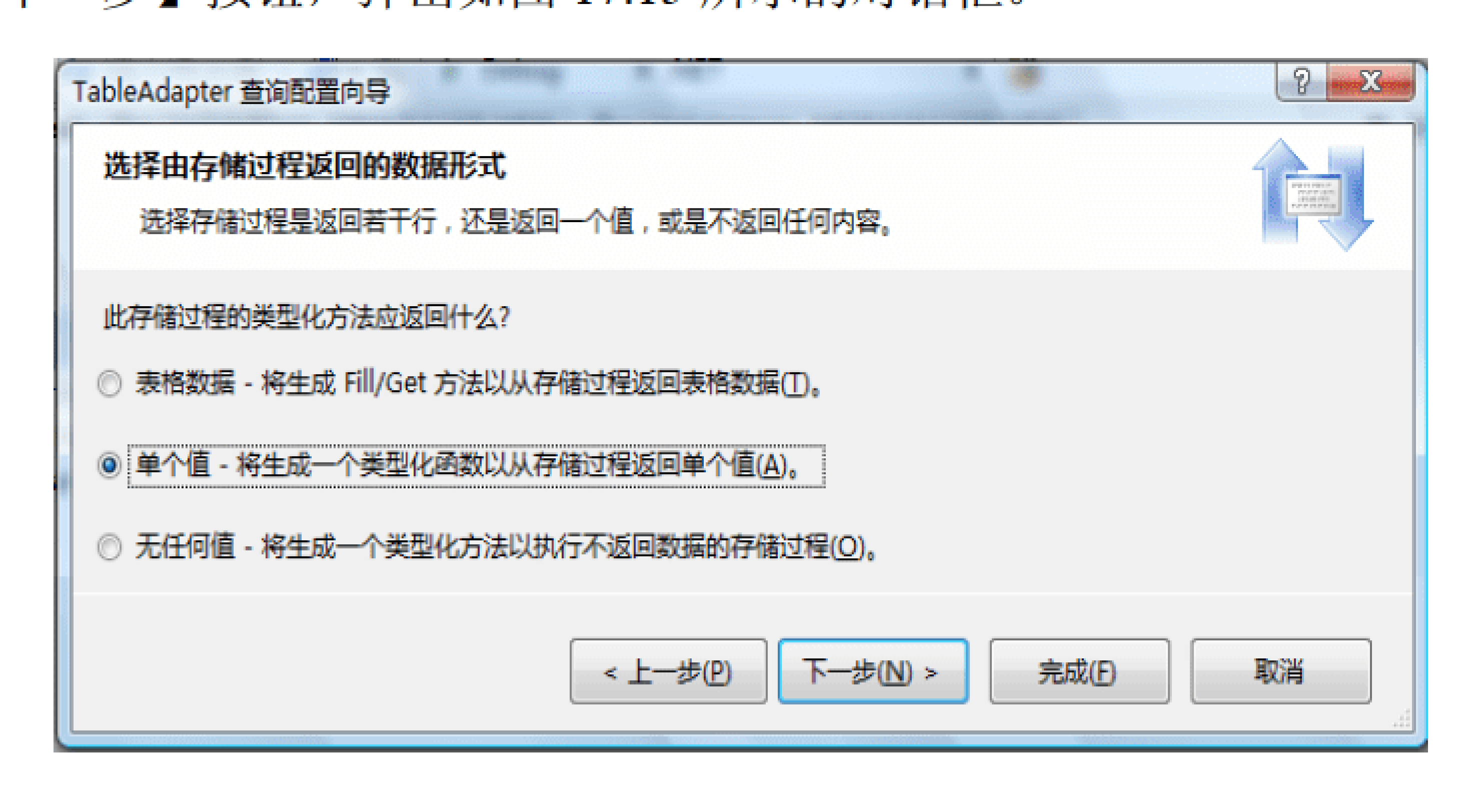


图 17.15　数据表适配器配置向导(3)

此对话框用于选择存储过程返回的数据形式。这里提供了三种选择：表格数据、单个值以及无任何值。

为了能够对取出的预付款做进一步处理，选择【单个值-将生成一个类型化函数以从存储过程返回单个值】单选按钮。此时的存储过程就相当于一个具有返回类型的函数。

单击【下一步】按钮，在弹出的对话框中确定函数名。此时，数据表适配器中就增加了一个新的方法。

按照同样的步骤右击【数据集】中的【数据表适配器】名，将修改预付款的存储过程增加进来。由于使用该存储过程的目的，只是用来修改客户表中的预付款字段。因此在图 17.15 中数据返回形式中选择【无任何值-将生成一个类型化方法以执行不返回数据的存储过程】单选按钮即可。其他步骤与前面所讲的相同。

此时，修改预付款的存储过程的界面如图 17.16 所示。

经过前面的配置以后在数据表适配器中已经增添了两个方法，如图 17.17 所示。

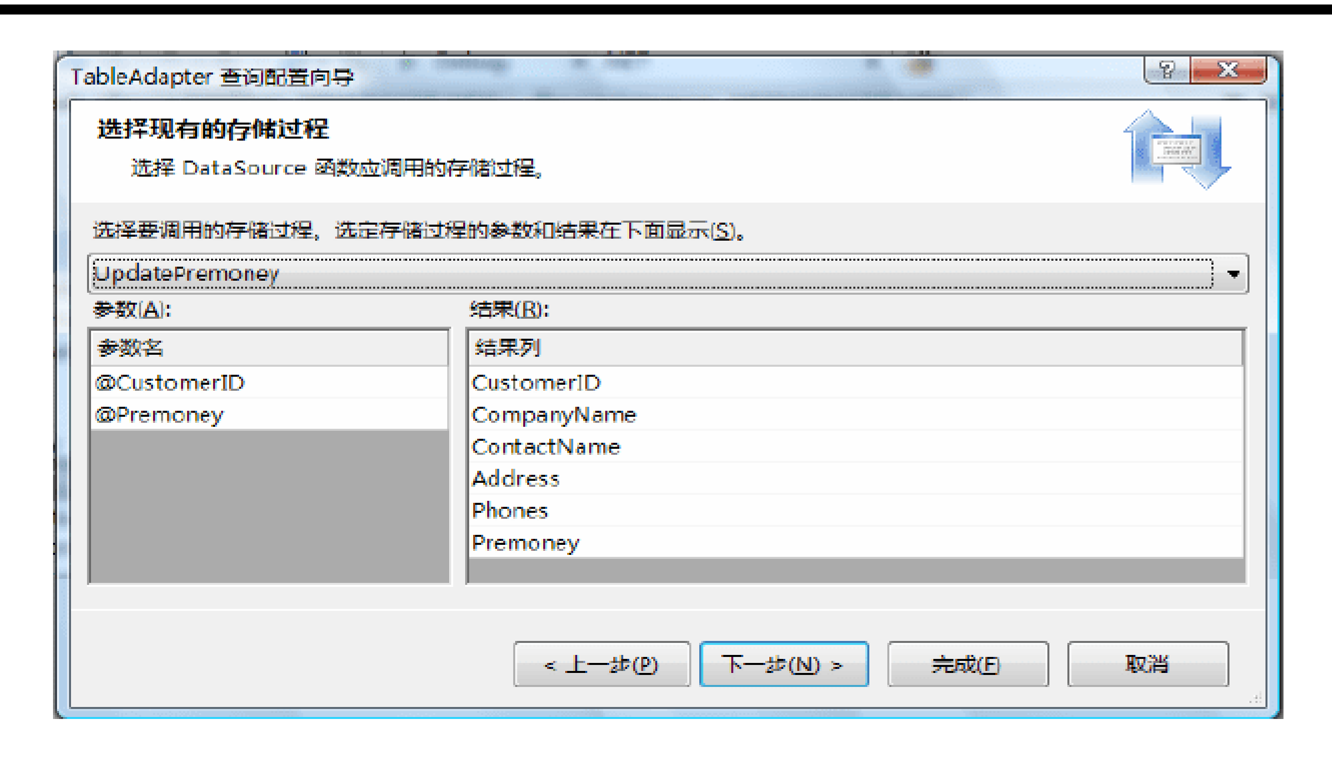


图 17.16　数据表适配器配置向导(4)

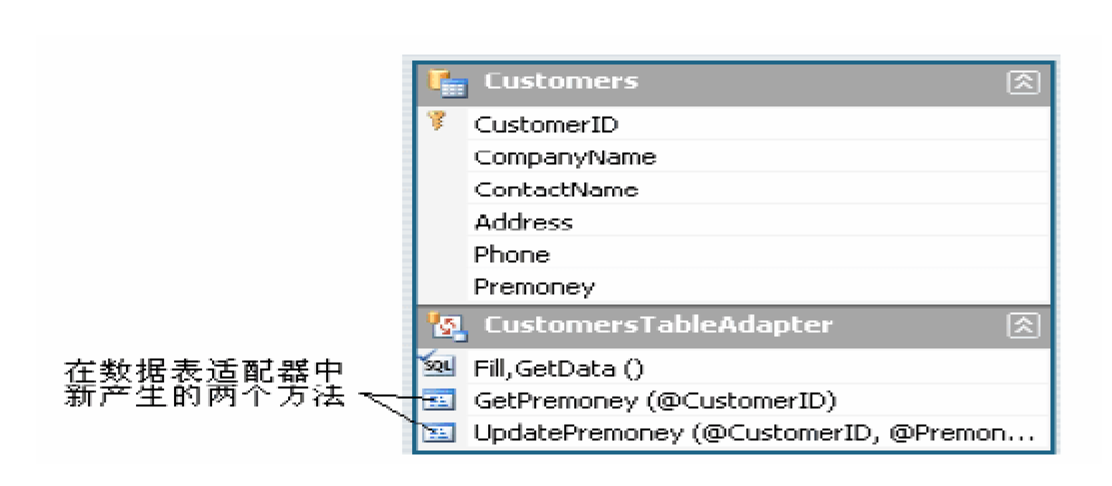


图 17.17　数据表适配器增加的方法

将上述设置保存并进行编译，当没有提示错误时再进行下一步。

(3)　创建自定义类。

右击网站名，选择【增加新项】后再选择【类】命令，以打开类的定义对话框。现在假定类名是 Class1.cs(注意：新创建的类必须放置在 App_Code 目录下)，类的定义如下。

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public class Class1
{
    DataSet1 ss = new DataSet1();                    //生成数据集对象
    DataSet1TableAdapters.CustomersTableAdapter tt; //声明数据表适配器
    public Class1()
    {
        tt = new DataSet1TableAdapters.CustomersTableAdapter();
        //生成数据表适配器对象 tt
    }

    public bool decreasePremoney(int cid, Decimal pay)//创建处理数据的方法
    {
        Decimal Prem = (Decimal)tt.GetPremoney(cid);  //取出预付款
        Decimal remain = Prem - pay;  //从预付款中减去本次需要的付款
```

```
            if (remain < 0)     //若预付款不够，返回 false
            {
                return false;
            }
            else                //将调整后的余额送回客户表，并返回 true
            {
                tt.UpdatePremoney(cid, remain);
                return true;
            }
        }
    }
```

这里的类处于显示层和数据层之间，它的一端从显示层获取输入的数据，另一端通过数据表适配器利用存储过程从数据表中获取数据。综合两方面的数据进行逻辑处理，并将处理结果分发到需要的地方。

Class1 类在这里的主要作用就是，将客户的预付款中减去本次交易的付款数，如果不够减，返回 false，表明此次交易失败；如果够减，返回 true，同时将减后的结果送回客户表的预付款字段中。

(4) 创建显示层。

在网页中创建显示界面，界面中的控件布局如图 17.18 所示。

客户ID 3300 需付款：1000

计算 是否成功 预付款修改成功。

CustomerID	CompanyName	ContactName	Address	Phone	Premoney
3300	河北宏泰公司	刘福贵	秦皇岛路径路43号	0335-64536783	2700.0000
3301	上海六福公司	李艳薄	上海常得路220号	021-88778899	300.0000
3302	长沙实用商贸公司	何时为	长沙51路235号	0731-77665544	80.0000
3303	北京前门商店	吴作为	北京前门饭店68号	010-77885544	7690.0000

图 17.18 预付款修改界面

其中 GridView1 控件可以通过 SqlDataSource 数据源控件直接与 Customers 数据表相连接，以显示客户表的数据。

网页中的代码如下。

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
public partial class premoney : System.Web.UI.Page
{
    Class1 ccss = new Class1();         //生成类对象
    protected void Page_Load(object sender, EventArgs e)
```

```
        {
        }
    //在按钮的 Click 事件中调用类方法以处理数据
        protected void Button1_Click(object sender, EventArgs e)
        {
            int cid = int.Parse(TextBox1.Text);          //客户标志
            decimal pay = decimal.Parse(TextBox2.Text); //本次交易需付款数
            bool result1=ccss.decreasePremoney(cid,pay);//调用类中的方法进行计算
            if (!result1)                               //如果预付款不够
                TextBox3.Text = "预付款不够。";
            else
                TextBox3.Text = "预付款修改成功。";       //处理成功
            GridView1.DataBind();
        }
    }
```

代码的关键部分在【计算】按钮的 Click 事件中，根据输入的客户 ID 以及本次交易需要的付款数，调用类(Class1)的方法来完成对预付款的修改工作。

17.6　小　　结

对于一个功能比较复杂或者功能有特殊要求的网站来说，最好采用三层架构，因为在中间层中可以放置任意定义的类，因此使用起来非常灵活，几乎不受限制。ASP.NET 3.5 为创建中间层提供了很多支持。对于访问数据库来说，最重要的支持是提供了对象数据源控件(ObjectDataSource)、数据集(DataSet)和数据表适配器(TableAdapter)。其中数据集是内存中的数据库，它可以在断开与数据源连接的条件下工作；数据表适配器是数据源与数据集之间的中间环节；显示控件通过对象数据源与数据集进行数据绑定。除之以外，对象数据源控件还能够在缓存数据、防止数据访问中的冲突等方面发挥重要作用。

17.7　习　　题

1. 填空题

(1) ASP.NET 3.5 中，中间层中的对象都应放入________目录中，以便各网页共享。

(2) ObjectDataSource 作为中间层的数据源控件，在访问数据库时能够提供以下几方面的支持______________、_______________、_________________。

2. 选择题

(1) 在三层架构中，客户端是系统的前台，负责客户界面的显示；后台负责数据的存储和管理；而中间层负责____________。

A. 非 UI 的逻辑处理　　B. 安全监督

C. 代码优化　　D. 协助后台管理

(2) 数据表适配器是数据集与数据源之间的桥梁，它的任务是__________。

A. 将检索后的数据填入数据集　　B. 将数据集中更新后的数据返回数据源

C. 传送客户输入的数据 D. A+B

(3) 在中间层中的数据集相当于________。

A. 内存中临时的数据库 B. 数据源中的数据表在内存中的副本

C. 客户输入的数据 D. 受保护的数据

3. 判断题

(1) 数据表适配器采用不同的检索语句，可以使数据集中获得不同的记录。 ( )

(2) 类库中的类才允许放入中间层中。 ( )

4. 简答题

(1) 创建中间层的目的是什么？三层架构适用于什么情况？

(2) ASP.NET 3.5 为创建中间层提供了哪些支持？

(3) 举例说明通过中间层访问数据库的步骤。

5. 操作题

(1) 创建一个中间层用来访问数据库。

(2) 直接调用中间层定义的类对象访问数据库。

(3) 通过 ObjectDataSource 数据源控件访问数据库。

(4) 用三层架构来设计一个网上舆论调查程序，要求在指定时间内防止客户重复投票，并能统计和显示投票的结果。题型参考例 9.4。

第 18 章　LINQ 技术

LINQ 是 ASP.NET 3.5 新推出来的一项突破性的技术创新，目的是用统一的方法来处理不同类型的数据，使应用程序与数据之间结合得更加紧密，处理起来更加流畅。本章将要讲述以下几个问题：

- 概述。
- LINQ 查询的语法基础。
- Lambda 表达式。
- LINQ to SQL。
- 利用 LINQ 编辑数据库。
- 使用 LINQ 数据源控件。
- 调用存储过程。
- 利用 LINQ 分析数据。

18.1　概　　述

LINQ 是.NET Language Integrated Query 的缩写，翻译成中文的意思是“.NET 语言集成查询”。

LINQ 的作用是什么？ C#.NET 的首席设计师 Anders Hejlsberg 概括地说，LINQ 的目标就是解决程序代码与数据间不匹配的问题。

计算机的主要任务就是处理数据，然而长期以来计算机的程序语言与数据处理之间并不完全匹配。各种主流程序语言，如 C++、C#、Java 等，经过长期的实践和发展，找到了“面向对象”这种最佳模式。而数据方面，不同类型的数据根据自己的目标，也都找到了适合于自身的模式。例如：

- 数据库建立在关系代数的基础之上，用表格及关系来组织数据，用 SQL 来处理数据。
- XML 用文档或文档对象模型(DOM)与标记相结合来组织数据，用 XPath 或 XQuery 来处理数据。
- 其他数据如 Excel、电子邮件信息或 Windows 注册表等，都有自己特有的组织形式和特定的应用程序接口(API)。

现在要通过应用程序来处理这些数据，就需要解决语言与数据之间模式不匹配的问题。如何解决这一问题呢？传统的方法就是在程序中先建立一个数据访问层，然后将各种数据处理的语句作为字符串嵌入到应用程序的代码中，处理时先映射到不同的模式中去，处理完毕后再返回来。这种方式带来了一些问题，例如：

- 设计人员必须掌握对各种不同类型数据的处理方法，这个要求增加了设计人员学习的难度。

- 嵌入式结合有时并不理想。以数据库为例，由于 SQL 属于后编译语言，语言中的问题往往只能在程序的运行中才能暴露，而不能在程序编译阶段发现。
- 应用程序对数据的处理过程需要在不同模式之间多次切换，因而进行得并不流畅。

现在 LINQ 技术就是要将不同数据的处理方法，统一到面向对象的模式下来，从而降低设计人员的学习难度，进一步提高数据处理的效率。

为此，首先要建立起各类数据与“类”和“对象”之间的映射关系。映射的过程虽然比较复杂，但系统可以提供工具来自动进行。LINQ 就是用来执行这项映射工作的，它的逻辑结构如图 18.1 所示。

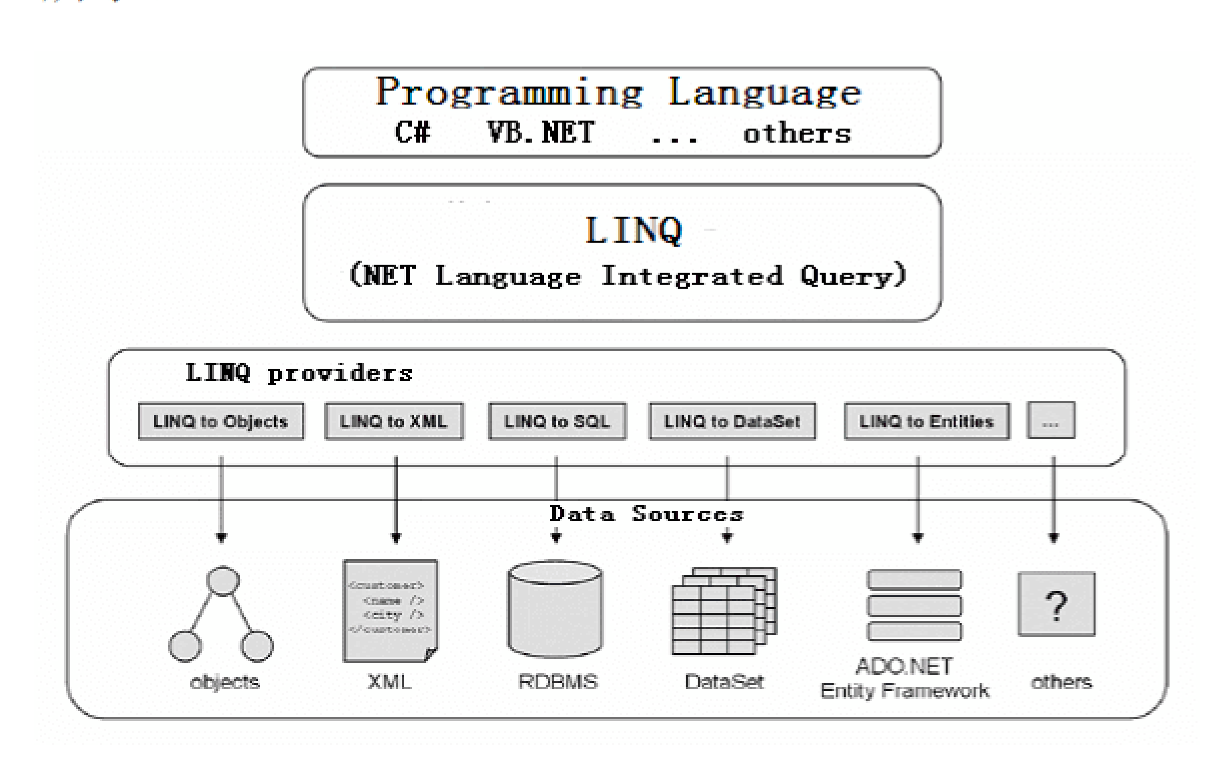


图 18.1 LINQ 的逻辑结构

图的最上层是应用程序，它使用各种程序语言；下面是各种不同类型的数据；中间是 LINQ 和 LINQ providers。在 LINQ providers 中包括 LINQ to Objects、LINQ to XML、LINQ to SQL、LINQ to DataSet 与 LINQ to Entities 等多个组成部分，分别能将不同类型的数据抽象到 LINQ 对象层来，然后再用面向对象的模式去处理它们。

LINQ 技术经过多年的研发(最先使用的名字是 NHibernate for .NET)，广泛地吸取了各种现代语言的精华，如强类型、现代动态语言、函数式语言等。研发的成果不仅能将各种类型的数据统一到面向对象的模式中来，而且还使 C#.NET 语言向前跨进了一大步。其中 PLINQ(也称并行 LINQ)就是 LINQ 技术的扩展，它允许在不单独处理线程、线程池及同步的情况下实现并行查询就是一例(由于篇幅限制本书不作介绍)。

注意： 目前 O/R 设计器是一个简单的对象关系映射器，它仅支持 1∶1 的映射关系，不支持复杂映射(例如，将实体类映射到联接表)。此外，该设计器还是一个单向代码生成器。这表示代码文件中只反映对设计器视图所做的更改。

18.2 LINQ 查询的语法基础

LINQ 语言是 C#.NET 3.0 语言的扩展(目前还能用于 Visual Basic 2008 语言)。下面将结合一个简单的示例来说明 LINQ 的语法结构。示例如下。

```
var st  =  from nn in person
              where nn.Length >=3
              select nn;
```

语句的作用是从一串人名(person)中找出长度大于或等于 3 的名字，将其值赋给变量 st。

整个语句中包括若干子句。

- from 子句：它定义了查询的数据源和局部变量。
- where 子句：它指明了对数据源筛选的条件。
- select 子句：它指明了查询的输出形式。

从上面的示例可以看出，LINQ 与 SQL 语句的格式相似，这是它的一大特点。在 LINQ 语句中将用到很多 SQL 子句。除上面三种以外，还将用到以下几种。

- group 和 into 子句：可用于由关键字分组返回结果。
- orderby 子句：可对查询结果按照升序或降序进行排列。
- join 子句：可以在成员的基础上连接不同的数据源。
- let 子句：它用于定义(或称“投影”)结果。在这里是返回满足条件的人名。

另外还将用到很多类似于 SQL 系统函数的方法。

- Average：取元素的平均值。
- Count：取元素的个数。
- Max：取元素中的最大值。
- Min：取元素中的最小值。
- Sum：取元素值的总和。

因此，学过 SQL 的读者可以利用已经具备的知识来编写 LINQ 语句。但是必须明确，尽管从形式上，LINQ 语句与 SQL 有很多相似之处，但是 LINQ 与 SQL 是两种完全不同性质的语言。下面就来比较两者之间的区别。

(1) 数据源不同。

首先需要搞清的是，上述示例中的 person 代表什么？

在 SQL 语句中它只能是数据库中的数据表。而 LINQ 可以应用于多种不同类型的数据，它可能来自数据库，也可能来自“字符串数组”，还可能来自内存中的数据集(DataSet)。只要对数据源加以定义，其他语句不需改变就能应用。

例如，当数据源是一个字符串数组(string[])时，定义的语句如下。

```
string[] BookName={"语文","数学","新闻学","英语","计算机科学","化学","伦理学"};
var course = from w in BookName
             where w.Length >2  // 字符串长度 > 2
             orderby w.ToUpper() descending   // 取大写后以降序排列
             select w;
course.Dump();    // 这是使用 LINQPad 工具时运行开始的语句
```

运行结果：

```
新闻学
伦理学
计算机科学
```

注意：LINQPad 是一个很好的学习 LINQ 的工具，它容量小，只有 2MB 左右，允许免费下载，下载后不需安装即可直接使用。下载网址是 http://www.linqpad.net/。

(2) 语法方面不同。

语法方面有以下几点重要的区别。

- 在 SQL 语句中总是将 Select 放在最前面，而 LINQ 将 from 放在最前面，在 from 与 select 之间允许放入 where、joins、orderby 或 into 等子句。
  为什么顺序上要做这样的改变？这是因为只有先用 from 放在前面确定了应用的范围，在后面的语句中才能更好地利用“智能提示”来选择需要的项目。
  另外，在 LINQ 语句以及变量中都是区分大小写的，而 SQL 语句中不区分大小写。
- 等号左边的变量 st(或 course)代表查询结果。这里用 var st(或 var course)的形式来声明变量。在 JavaScript 语言中，经常会遇见这种形式，但这里与 JavaScript 的变量声明不同，JavaScript 不是强类型语言，而 st(或 course)是一个强类型变量，它的类型是什么，这里虽然没有明确指出，但编译器会根据等号右边的表达式来自动推断出它的类型(例如，当数据源是字符串数组时，它就是 Ienumerable <string>类型)。这是 C# 3.0 语言新提供的“类型推断”功能，它允许使用 var 关键字，具体类型则由编译器自动推断出来。这也就为设计者省去了很多麻烦，也减少了很多可能产生的错误。
- 语句中的 nn(或 w)代表什么。它是一个局部变量，在 from、let、join 或 into 中用来表示某个范围。这个变量的类型虽然没有直接定义，但是它的类型也由编译器自动确定。

18.3 Lambda 表达式

在 C# 3.0 语言中引入了 Lambda 表达式(Expression)。Lambda 是一种定义匿名方法的表达式，可以被嵌入到 LINQ 中使用，它结构简练，却可以包含丰富的内容。Lambda Expression 的格式如下：

```
(参数列表) => 表达式或者语句块
```

表达式中以“=>”为界(“=>”念成 goes to)，分为左右两部分，左边是参数列表，右边是函数的实现部分(函数体)。在参数列表中可以包括 0 到多个参数。参数的类型可以隐含也可以直接显示。隐含时变量属于什么类型，由编译器根据表达式的前后文来自动确定。

例如，下列 Lambda 表达式定义了一个匿名方法。

```
(int x, int y) => (x*y) / 2 ;
```

用传统的函数来表达上述 Lambda 表达式时为

```
int f (int x, int y)
 {
   return (x * y)/ 2 ;
 }
```

Lambda 还可表达为一棵“目录树”。所谓目录树就是由操作数和操作符节点组成的树。上述 Lambda 的目录树如图 18.2 所示。

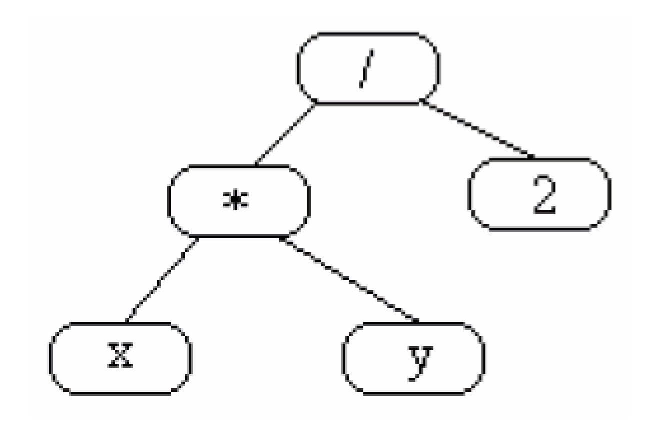


图 18.2　Lambda 的目录树

几个典型的 Lambda 表达式示例如下。

(1)　显示类型的输入参数：

```
(int x) => x + 1
(int x) => { return x + 1; }
```

(2)　隐含类型的输入参数：

```
x => x + 1             //一个参数
x => { return x + 1; }
customer => customer.Name
person => person.City == "Paris"
(x, y) => x * y      //两个或多个参数
(person, minAge) => person.Age >= minAge
```

(3)　无输入参数：

```
( ) => 1
( ) => Console.WriteLine()
```

下面用几个示例来说明 Lambda 表达式的使用方法。

例 18.1　计算两个整型数相乘。

```
static void Main()
{
  Func<int, int, int> mul =(int x, int y) => x * y;
  Console.WriteLine(mul(6, 4));
  Console.ReadLine();
}
```

其中 Func<int, int, int>代表 Func<T,T,TResult>。前两项代表输入参数的类型，第三项为返回类型。以上代码运行的结果为 24。

第一条语句也可简化为

```
Func<int, int, int> mul =( x, y) => x * y;
```

例 18.2　输出包括‘国’字与‘中’字的字串。

```
static void Main()
{
 var words = new string[]{"中国", "英国", "法国","中国人"};
 IEnumerable<string> hasDAndE =
       words.Where(s => s.Contains('国') && s.Contains('中'));
```

```
    foreach(string s in hasDAndE)
      Console.WriteLine(s);
      Console.ReadLine();
}
```

运行结果如下:

```
中国
中国人
```

例 18.3 将多个整型数按照由小到大排列显示。

```
static void Main()
{
  var numbers = new int[] {1, 3, 5, 7, 9, 2, 4, 6, 8, 10 };
  IEnumerable<int> ordered = numbers.OrderBy(n=>n);
  foreach(var num in ordered)
  Console.WriteLine(num);
  Console.ReadLine();
}
```

运行结果如下:

数字按照由小到大排列

例 18.4 确定给定的字符串中是否包含"机"字。

```
static void Main()
{
 string findword = "机";
 var words = new string[]{
      "电视", "洗衣机", "电冰箱", "手机"};
 var matches = words.Select(s => s.Contains(findword));
 foreach(var str in matches)
      Console.WriteLine(str);
 Console.ReadLine();
}
```

其结果显示:

```
False
True
False
True
```

如果要显示出具体名称时，语句应改写如下。

```
static void Main()
{
 string findword = "机";
 var words = new string[]{
       "电视", "洗衣机", "电冰箱", "手机"};
IEnumerable<string> matches = words.Where(s => s.Contains(findword));
 foreach(var str in matches)
       Console.WriteLine(str);
 Console.ReadLine();
}
```

运行结果如下:

```
洗衣机
手机
```

18.4　LINQ to SQL

前面已经介绍 LINQ 的语言基础，下面重点讲述利用 LINQ 访问 SQL Server 数据库的方法。

数据库是用表格中的行、列和关系来表示数据的。为了用 LINQ 来处理数据库中的数据，须先将数据库映射成类和对象。它们之间的映射关系是，每个类对应于一个数据行，每个类的属性对应于一个数据列。

映射的工具有两种：SQL Metal 和 LINQ to SQL。其中 SQL Metal 是一个命令行工具，而 LINQ to SQL 是一个 O/R 可视化设计器。下面将重点介绍后者。

18.4.1　将数据库映射成类和对象

对象/关系设计器(O/R 设计器)提供了一个可视化的设计界面，用于建立数据库与实体类之间的映射。就是说，通过 O/R 设计器可以在应用程序中创建数据库的对象模型。

目前，O/R 设计器只支持 SQL Server 2000/2005/2008/Express 或 Compact 3.5 数据库的映射。

现在以微软提供的 Northwind 样板数据库为例，来讲解使用 O/R 设计器的方法。

(1)　在网站的 Add_Data 目录下将 Northwind.mdf 数据库调过来。

(2)　在【添加新项】对话框中选择【LINQ to SQL 类】。在打开的对话框中修改文件名(这里改为 Northwind.dbml)，语言选择 Visual C#。最后在系统的提示下将映射的类放在 App_Code 目录下。此时将打开一个可视化的界面。

(3)　界面分为左、右两部分。左边可放入数据表及它们之间的关系；右边(单击鼠标右键后可在隐藏、显示之间切换)放入存储过程。现在打开数据库资源管理器，将多个数据表拖到左边的窗格中来，然后将若干存储过程拖入到右边窗格中。其结果如图 18.3 所示。

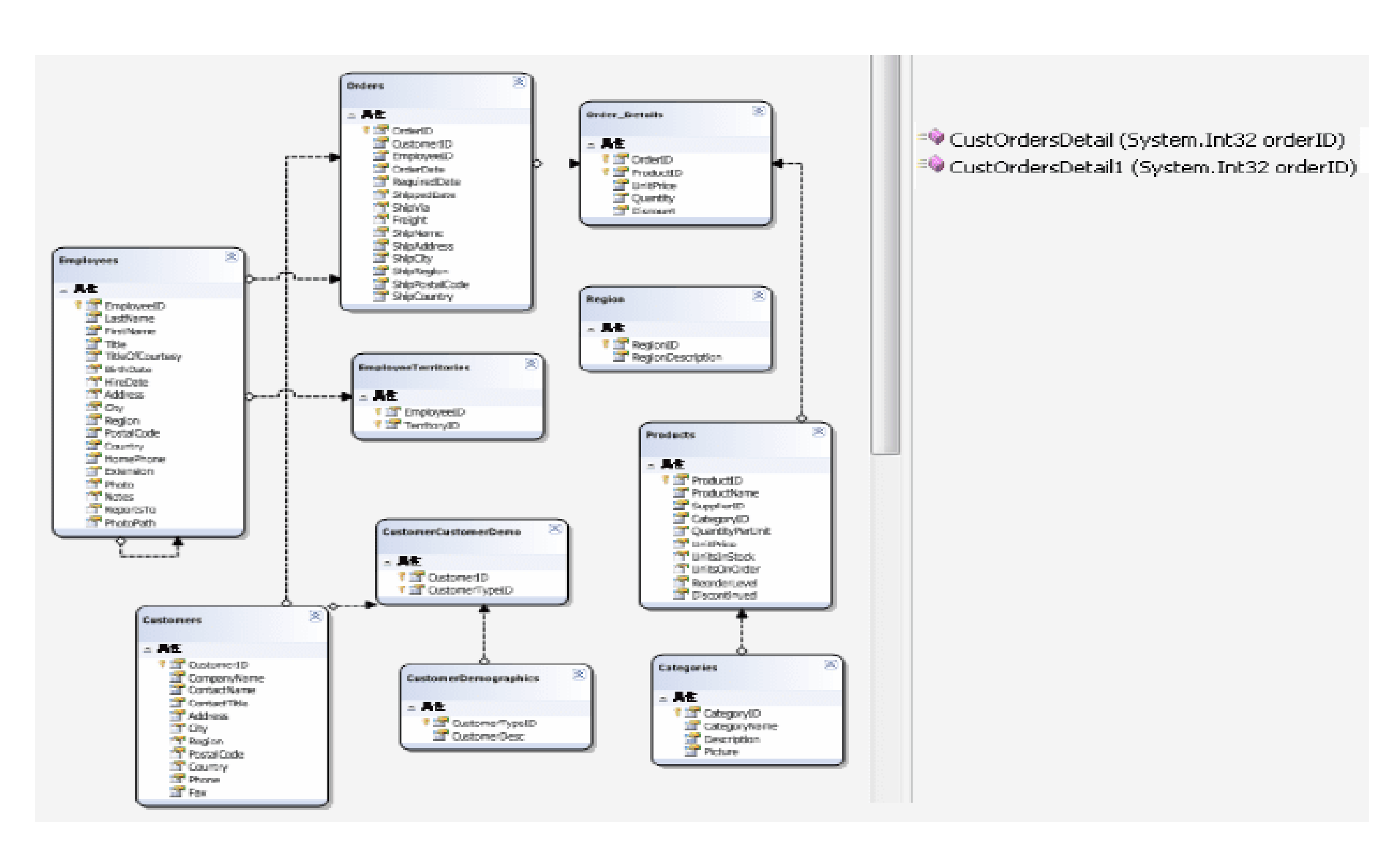


图 18.3　LINQ to SQL 可视化界面

界面中的方框代表数据表映射后的实体类，数据库中的字段就映射成实体类中的属性。实体类之间的连线(Associations)代表数据表之间的关系(Relationships)。箭头代表“一对多”的同步关系，箭头的起始端代表“一”端，终点代表“多”端。

(4) 启动编译以完成映射工作。

18.4.2 映射中的对应关系

数据库与实体类之间的映射关系是：

(1) 每张数据表(Table)映射成一个实体类(Entity Class)。

(2) 数据表中的字段(Column)映射成类中的属性(Class Member)。

(3) 数据库的关系(Relationships)映射成类的关联(Association)。

(4) 存储过程(Stored Procedure)映射成类中的方法(Method)。

(5) 除此之外，系统还生成了一个 DataContext 类(这里为 NorthwindDataContext 类)。在 DataContext 类中定义了 LINQ 与外部关系数据库之间的联系，如同一座桥梁架设在实体类与数据库之间。其具体作用包括：

- 连接数据库。
- 从数据库中读取数据。
- 向数据库提交编辑后的结果。

经过上述映射后，在 App_Code 目录下将自动生成一个.dbml 文件(包括.dbml.layout 和.designer.cs 两个文件)。

双击 dbml 文件时可以看到可视界面。右击.dbml 文件名，在弹出的快捷菜单中选择【打开方式】命令，然后选择【XML 编辑器】时，可以看到.dbml 文件的内容。这是一个 XML 文件，它用 XML 标签定义了数据表(Table)、数据列(Column)的结构以及各表之间的关系(Association)。

双击打开.designer.cs 文件时，可以看到用 C#语言定义的类名，以及类包括的属性和方法。其中也可以找到 DataContext 类的类名。

映射后可以在各个实体类的属性对话框中修改该类的属性，属性对话框如图 18.4 所示。

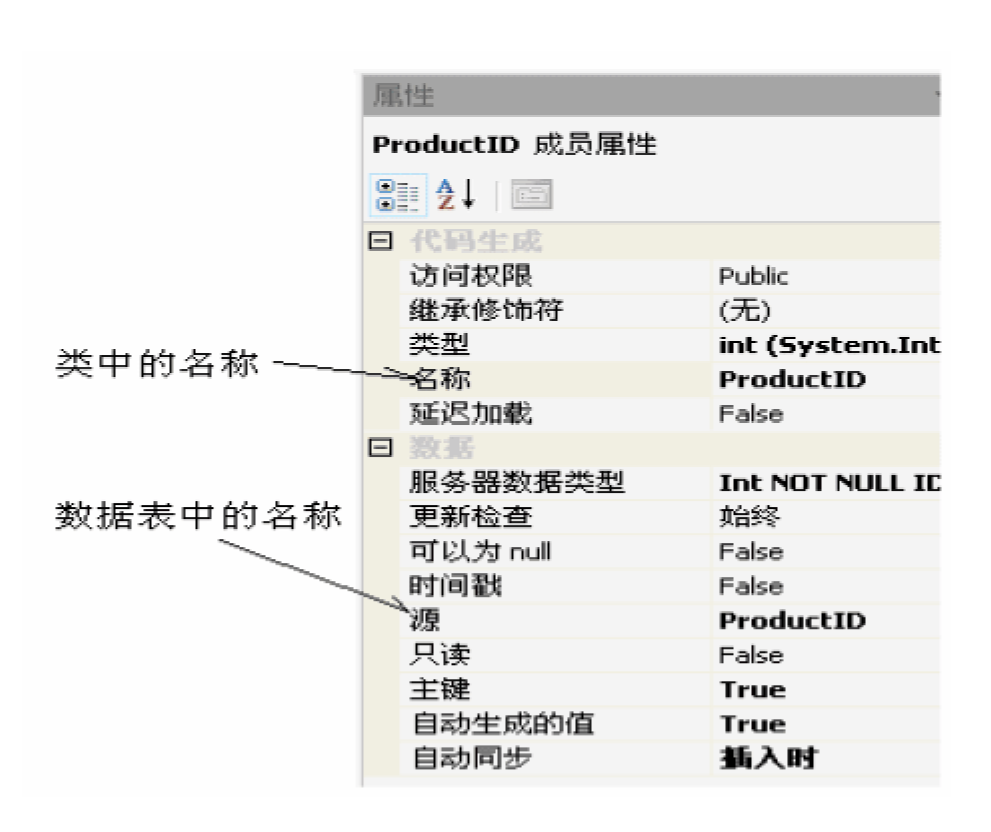


图 18.4 映射后的属性对话框

属性对话框的上半部是类成员的属性，下半部是数据库成员的属性。例如：

- 上面的“名称”是类中的属性名，下面的“源”是对应于数据库中的字段名。如果将“名称”改用汉字时，网页中将用汉字显示相应的字段名。
- 如果将属性的【延迟加载】设为 true 时，该字段只有在以后使用时再加载。一些容量大的成员(如较大的图片)，可以利用这个属性缩短打开网页的时间。
- 在视图中右击类的属性(字段)后在弹出的快捷菜单中选择【删除】命令，可以去掉一些应用程序中不用的字段(在数据库中字段并没有消失)。

18.4.3　映射后的部分代码

打开 Northwind.designer.cs 设计文件，就可以看到映射后各实体类的代码。

以 Products 数据表为例，它已经映射成以下的类。

```
[Table(Name="dbo.Products")]
     public partial class Products :…
     {…}
```

字段如 ProductID、ProductName 都已经映射成以下的属性。

```
private int _ProductID;
     private string _ProductName;
public int ProductID
     {
      get
         {
           return this._ProductID;
         }
         set
         {
           if ((this._ProductID != value))
              {
                …
              }
         }
    }

    [Column(Storage="_ProductName", DbType="NVarChar(40) NOT NULL",
CanBeNull=false)]
    public string ProductName
    {
      get
       {
         return this._ProductName;
       }
      set
       {
         if ((this._ProductName != value))
          {
            …
          }
       }
    }
```

18.4.4 数据库显示和查询

1. 利用 LINQ 显示数据表

下面将结合 GridView 控件(或其他控件)，用编写 LINQ 代码的方法来显示数据表中的数据。例如：

```
protected void Page_Load(object sender, EventArgs e)
     {
      NorthwindDataContext db = new NorthwindDataContext();
      var product = from p in db.Products
                    select new
                    {
                      p.CategoryID,
                      p.ProductID,
                      p.ProductName,
                      p.QuantityPerUnit,
                      p.UnitPrice
                    };
      GridView1.DataSource = product;
      GridView1.DataBind();
   }
```

上述代码可以显示 Products 数据表中的 4 个字段。

如果记录数较多时，可以对记录进行分页。分页时除需设置允许分页，指定每页包括的记录条数以外，还需在 GridView 控件的 PageIndexChanging 事件中设置以下代码。

```
 protected void GridView1_PageIndexChanging(object sender,
GridViewPageEventArgs e)
   {
     GridView1.PageIndex = e.NewPageIndex;
     DataBind();
 }
```

如果想分组显示时，其代码如下。

```
protected void Button2_Click(object sender, EventArgs e)
{
NorthwindDataContext db = new NorthwindDataContext();
  var q = from p in db.Products
       group p by p.CategoryID into g
       // 按 CategoryID 进行分组
       select new
          {
           类型 ID = g.Key,
           总和 = g.Sum(p => p.UnitPrice),  //显示每组的总金额
           最低价 = g.Min(p => p.UnitPrice),//显示每组中的最低价
           最高价 = g.Max(p => p.UnitPrice) //显示每组中的最高价
          };
         GridView1.DataSource = q;
     GridView1.DataBind();
}
```

2. 跨数据表查询

如果数据表之间已经建立了关系，则可以按照下列办法跨数据表进行查询，并将多个数据表中的查询结果显示在同一张表格中。

例如产品类型表(Categories)与产品表(Products)已经建立了“一对多”的同步关系，现在想要将“类型名”与相关的“产品名”显示到同一张表格中。由于类型名与产品名分别放在不同的数据表中，因此应该使用跨数据表查询的方法。其代码如下。

```
protected void Button2_Click(object sender, EventArgs e)
    {
    NorthwindDataContext db = new NorthwindDataContext();
       var q = from c in db.Categories
               from p in c.Products
               where c.CategoryID == int.Parse(DropDownList1.SelectedValue)
               select new { c.CategoryName, p.ProductName };
        GridView1.DataSource = q;
        GridView1.DataBind();
    }
```

代码中用下拉列表控件中选择项的值(SelectedValue)作为条件，显示“类型名”与对应的“产品名”。

上述代码也可以改写如下。

```
protected void Button1_Click(object sender, EventArgs e)
    {
        NorthwindDataContext db = new NorthwindDataContext();
        var q = from c in db.Products
                where c.Categories.CategoryID ==
int.Parse(DropDownList1.SelectedValue)
                select new { c.Categories.CategoryName, c.ProductName };

        GridView1.DataSource = q;
        GridView1.DataBind();
    }
```

在后面这段代码中直接从同步的“多”方进入。

```
from c in db.Products
```

因而可以通过它找到“一”方的数据表(实体类)，并调用该表的字段(属性)；与此相反，如果从“一”方进入不能直接找到“多”方的数据表，只能通过另一个 from 语句进行设置。

18.5　利用 LINQ 编辑数据库

数据库(表)的编辑包括更新(Updating)、插入(Inserting)、删除(Deleting)三方面。在更新和删除中都需要先定位，再执行编辑操作。

编辑工作都是先在映射的实体类中进行，完成后再将结果用 SubmitChanges() 方法送回数据库。如果没有最后这一句，数据库中的数据都不会改动。

下面将围绕 Northwind 数据库来讲解数据库的编辑方法。网页界面如图 18.5 所示。

类型ID 2 产品ID 修改后的单价 产品名：
按照类型查询 修改单价 插入新纪录 删除纪录

类型ID	产品ID	产品名	单元数量	单价
2	3	盐	每箱10公斤	10.0000
2	4	麻油	每箱20瓶	22.0000
2	5	胡椒粉	每箱10公斤	21.3500
2	6	味精	每箱10公斤	25.0000
2	8	酱油	每箱20瓶	40.0000
2	15	湖南辣椒粉	每箱10公斤	15.5000
2	44	四川辣椒粉	每箱10公斤	29.4000

图 18.5　网页界面

18.5.1　更新数据表(Updating)

代码如下。

```
protected void Button2_Click(object sender, EventArgs e)
{
  NorthwindDataContext db = new NorthwindDataContext();
      var q = from p in db.Products
             where p.产品 ID == int.Parse(TextBox8.Text)
             select p;
//先定位需要修改的记录
       foreach (var c in q)
       {
         c.单价=decimal.Parse(TextBox3.Text);
       }
        db.SubmitChanges();
     // 返回数据库进行修改
        }
```

如果改用 Lambda 表达式，可以采用更加简洁的形式取得同样的效果。代码如下。

```
protected void Button2_Click(object sender, EventArgs e)
{
       NorthwindDataContext db = new NorthwindDataContext();
       Products updatePrice=db.Products.Single(m=>m.产品
ID==int.Parse(TextBox8.Text));
        //先定位需要修改的记录
       updatePrice.单价=decimal.Parse(TextBox3.Text);
        //指定字段修改的值
        db.SubmitChanges();
        //返回数据库修改
      }
```

18.5.2　插入新记录(Inserting)

增添记录时需首先创建一条新记录(对象)，包括数据表中不能为 null 的所有字段；然后再将这个对象作为属性成员添加到原有的实体类中，最后增加到数据库中。代码如下。

```
    protected void Button3_Click(object sender, EventArgs e)
```

```
{
NorthwindDataContext db = new NorthwindDataContext();
      Products pp = new Products { 类型 ID = int.Parse(TextBox1.Text),
                                  产品名 = TextBox4.Text };
      //先生成一条新记录
       db.Products.InsertOnSubmit(pp);
       //将新记录加入到类中
       db.SubmitChanges();
       //将新记录增加到数据库中
     }
}
```

18.5.3　删除记录(Deleting)

删除记录前必须定位，找到准备删除的记录，然后再执行删除操作。代码如下。

```
protected void Button4_Click(object sender, EventArgs e)
    {
     var q = from p in db.Products
            where p.产品 ID == int.Parse(TextBox8.Text)
            select p;
     foreach (var c in q)
       {
          db.Products.DeleteOnSubmit(c);
        }
       db.SubmitChanges();
    }
```

也可以采用另一种更加简洁的形式，代码如下。

```
 protected void Button4_Click(object sender, EventArgs e)
    {
       Products DD=db.Products.Single(m=>m.产品
ID==int.Parse(TextBox8.Text));
       db.Products.DeleteOnSubmit(DD);
       db.SubmitChanges();
     }
```

18.6　使用 LINQ 数据源控件

ASP.NET 3.5 提供了 LinqDataSource 控件。该控件的作用是将多种数据显示控件，如 GridView、Repeater、DataList、ListView、DetailsView 等，与由 LINQ to SQL 或 LINQ to Object 创建的实体类进行数据绑定，以便完成对数据的检索(选择、筛选、分组、排序)和编辑(更新、删除和插入)工作。利用 LINQ 数据源控件显示或编辑数据库非常简单，但灵活性不如直接编写代码。

下面以 GridView 显示控件为例，讲解通过 LINQ 数据源控件查询数据库和管理数据库的方法。

(1) 在网站中放入 Northwind 数据库和 GridView1 控件，在控件的属性对话框中可以设置各种属性(如分页等)。

(2) 右击 GridView1 右上角标签，选择 LINQ 数据源控件，并打开【配置数据源】对话框。配置 LinqDataSource 数据源的过程与配置 SqlDataSource 基本相同。

(3) 单击【完成】按钮即完成配置工作。

18.7 调用存储过程

下面用一个示例来说明在 LINQ to SQL 中调用存储过程的方法。

假定在 O/R 设计视图中放入一个存储过程，名为 Sales_by_Year。原来的存储过程代码如下。

```
ALTER procedure "Sales by Year"
    @Beginning_Date DateTime, @Ending_Date DateTime AS
SELECT Orders.ShippedDate, Orders.OrderID, "Order Subtotals".Subtotal,
DATENAME(yy,ShippedDate) AS Year
FROM Orders INNER JOIN "Order Subtotals" ON Orders.OrderID = "Order
Subtotals".OrderID
WHERE Orders.ShippedDate Between @Beginning_Date And @Ending_Date
```

从代码可以看出，这个存储过程将返回在指定时间范围内的订单情况。经过 O/R 设计视图的映射，存储过程变成为方法，名字与存储过程相同。

为调用这个方法，在网页界面上先放两个 TextBox 控件，分别用于输入时间和截止时间。另外放一个按钮，其 Click 事件的代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
  {
    NorthwindDataContext db = new NorthwindDataContext();
    var order =
db.Sales_by_Year( DateTime.Parse(TextBox1.Text),DateTime.Parse(TextBox8.
Text));
        //调用方法
    GridView1.DataSource = order;
    GridView1.DataBind();
    }
```

18.8 利用 LINQ 分析数据

下面列举几个常用的应用示例，来说明 LINQ 进行数据分析的方法。示例中准备使用 Wizard 控件将多种分析综合到一个网页中。使用的界面如图 18.6 所示。



图 18.6 利用 Wizard 控件进行综合分析

18.8.1　销售分析

在一定的时间范围内，统计按月销售的金额。

界面中先放入两个 TextBox 以便输入统计日期的范围，格式为：月/日/年。再放入一个按钮，其 Click 事件的代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
    {
      NorthwindDataContext db = new NorthwindDataContext();
      var q = from p in db.Order_Details
              where p.Orders.OrderDate >= DateTime.Parse(TextBox1.Text) &&
                    p.Orders.OrderDate <= DateTime.Parse(TextBox8.Text)
              group p by p.Orders.OrderDate.Value.Month into g
              orderby g.Sum(x => x.Quantity * x.UnitPrice) descending
              select new
             {
                月份 = g.Key,
                金额 = g.Sum(x => x.Quantity * x.UnitPrice)
             };
            GridView1.DataSource = q;
            GridView1.DataBind();
      }
```

18.8.2　对产品销路的分析

在一定时间范围内，统计各产品的销售情况，按照销售量的降序进行排列。

界面中先放入两个 TextBox 控件以便输入统计的日期范围，格式为：月/日/年。再放入一个按钮，其 Click 事件的代码如下。

```
  protected void Button1_Click(object sender, EventArgs e)
    {
      NorthwindDataContext db = new NorthwindDataContext();
      var q = from p in db.Order_Details
              where p.Orders.OrderDate >= DateTime.Parse(TextBox1.Text)
                    &&p.Orders.OrderDate <= DateTime.Parse(TextBox8.Text)
              group p by p.Products.ProductName into g
              orderby g.Sum(x => x.Quantity * x.UnitPrice) descending
              select new
             {
                 产品名 = g.Key,
                 金额 = g.Sum(x => x.Quantity * x.UnitPrice)
             };
       GridView1.DataSource = q;
       GridView1.DataBind();
    }
```

18.8.3　职工管理

1. 职工的业绩比较

如果职工业绩可以用该职工经手的合同金额来衡量。其代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
```

```
    {
      NorthwindDataContext db = new NorthwindDataContext();
      var q = from p in db.Order_Details
              where p.Orders.OrderDate >= DateTime.Parse(TextBox1.Text) &&
                   p.Orders.OrderDate <= DateTime.Parse(TextBox8.Text)
              group p by p.Orders.EmployeeID into g
              orderby g.Sum(x => x.Quantity * x.UnitPrice) descending
              select new
             {
                职工标志 = g.Key,
                金额 = g.Sum(x => x.Quantity * x.UnitPrice)
             };
       GridView1.DataSource = q;
       GridView1.DataBind();
    }
```

2. 显示当天生日的职工

代码如下。

```
protected void Button3_Click(object sender, EventArgs e)
{
 NorthwindDataContext db = new NorthwindDataContext();
 var birth = from b in db.Employees
   where b.BirthDate.Value.DayOfYear == System.DateTime.Now.DayOfYear
     select new
      {
        b.EmployeeID,
        b.BirthDate,
        b.FirstName,
        b.LastName,
        b.TitleOfCourtesy
       };
        GridView3.DataSource = birth;
        GridView3.DataBind();
  }
```

18.8.4 批量修改数据

例如所有的水果降价 10%，按照原来价格的 90%出售。可以用下列代码来实现。

```
protected void Button2_Click(object sender, EventArgs e)
    {
     NorthwindDataContext db = new NorthwindDataContext();
     var q = from p in db.Products
           where p.Categories.CategoryName == "水果"
           select p;
       foreach (var c in q)
       {
           c.单价 = c.单价 * 0.9M;   //类型为 money 时，后面都要加 M
           db.SubmitChanges();
       }
```

18.9　小　结

LINQ 的主要目标是，将应用程序对各种类型的数据处理统一到面向对象的模式中来。从外观上来看，LINQ 的格式与 SQL 语句相似，但它们是两种不同的语言，其本质的区别在于数据源不同。SQL 的数据源只能是数据库(表)，而 LINQ 的数据源可以来自多种不同类型。其他方面还有一些重要的区别。比如 LINQ 是一种强类型语言，其类型采用了“类型推断”技术，即由编译器根据前后文的情况自动确定类型。LINQ 区分大小写，而且语言的顺序也有所不同。在 LINQ 语言中嵌入 Lambda 表达式是 C# 3.0 的重要发展，Lambda 具有高度抽象和很强的表达能力，嵌入在 LINQ 语句中能够用更加简练的语句实现更强的功能。

LINQ to SQL 是 LINQ 应用的一个重要方面。它首先利用可视化工具(O/R 设计器)将数据库(表)映射到实体类中来。其中 DataContext 类在实体类与数据库之间起桥梁作用。然后按照与应用程序的其他语言相同的方式来显示、编辑和分析，使应用程序与数据之间更紧密地结合在一起，最后通过 DataContext 将编辑的数据返回数据库。

18.10　习　题

1. 填空题

(1) Lambda 是一种定义__________的表达式。

(2) 在 LINQ providers 中包括__________、__________、__________、与__________、__________多个组成部分，分别能将不同类型的数据抽象到 LINQ 对象层来。

2. 选择题

(1) LINQ 与 SQL 的本质区别是__________。

A. 语法不同　　B. 安全程度不同

C. 数据源不同　　D. 是否强类型变量

(2) 在 LINQ 语言中要将 from 放在前面，是因为__________。

A. 与 SQL 相区别　　B. 集中所有数据

C. 安全因素　　D. 能够利用智能提示来选项

3. 判断题

(1) LINQ 不是一种强类型语言。　(　)

(2) LINQ 语言的目标是用面向对象方式处理各种类型的数据。　(　)

(3) Lambda 是一种定义匿名方法的表达式。　(　)

4. 简答题

(1) 在 LINQ 语言中什么是“类型推断”？举例说明。

(2) 在类的映射中 DataContext 类的作用是什么?

(3) 将下面的方法翻译成 Lambda 表达式。

```
double  f (double  x, double  y)
  {
       turn (x * y)/ 2 ;
  }
```

5. 操作题

建立一个学生成绩管理数据库，包括 10 名同学的 5 门功课成绩。

(1) 利用 LINQ 来显示和编辑学生成绩管理库。

(2) 利用 LINQ 来分析学生学习成绩。

第四部分

母版页与角色管理

前几章讲述的重点是单个网页的设计。这一部分将从网站的全局上来讲述以下 4 方面的问题:

- 网站的显示风格。
- 网站导航。
- 基于角色的安全技术。
- 网站的个性化服务。

第 19 章　主题、用户控件和母版页

本章将首先讲述网站显示风格的设计。随着网站功能的增强，网站逐渐变得庞大起来。现在一个网站包括几十、上百张网页已是常事。这种情况下，如何简化对众多网页的设计和维护，特别是如何解决好对一批具有同一风格网页界面的设计和维护，就成为比较普遍的难题。ASP.NET 提供的主题、用户控件和母版页技术，为控件的外貌、网页的局部再到全局风格的一致提供了最佳的设计方案。本章将讲述这三种技术，具体包括：

- 主题。
- 用户控件。
- 母版页。

19.1　主　　题

19.1.1　什么是主题

主题是一个目录，这个目录中只允许放进三种类型的文件：皮肤文件(后缀为.skin 的文件)、样式文件(后缀为.css 的文件)和一些图像文件。

皮肤文件(又称外观文件)用来定义一批服务器控件的外貌，级联样式表用来定义HTML 的标签。由皮肤、样式再加上相关的图像组成的主题实际上代表着一种显示风格。

主题目录必须放置在 App_Themes 专用目录之下，它们的关系如图 19.1 所示。

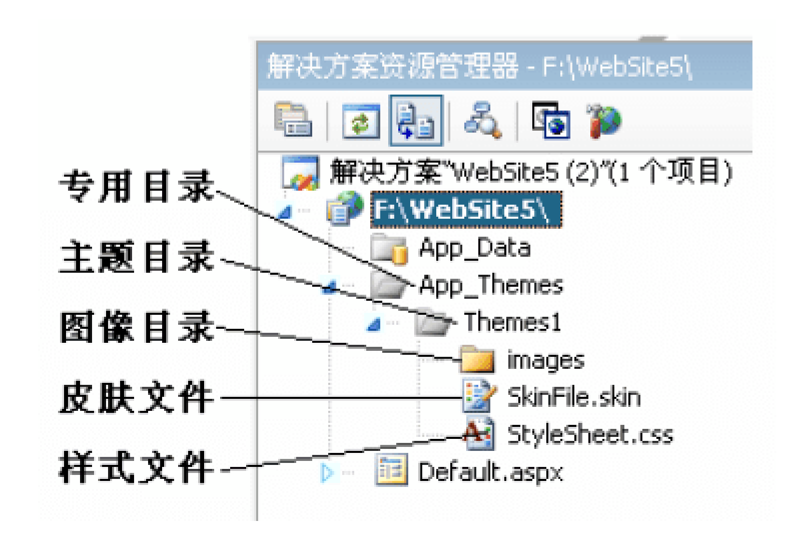


图 19.1　专用目录、主题目录及其包含的文件

每个 App_Themes 专用目录下可以设置多个主题目录。当 App_Themes 专用目录下包括有多个主题目录时，就意味着网站可以根据需要选择不同的风格来显示网页。

19.1.2　创建主题及皮肤文件的方法

下面介绍主题及皮肤文件的创建方法。

(1)　右击网站名，选择【添加文件夹】命令，然后选择【主题文件夹】项，系统将会在应用程序的根目录下自动生成一个专用目录 App_Themes，并且在这个专用目录下放置

主题文件夹。这里给文件夹取名为 Themes1。

(2)　右击主题文件夹名，在弹出的菜单中选择【外观文件】命令。

(3)　弹出一个皮肤文件的框架，可以给该文件改名，但是文件的后缀必须是.skin。这里取名为 SkinFile.skin。

(4)　在皮肤文件(这里是 SkinFile.skin)中给 TextBox 和 Button 两种控件定义外貌的语句如下。

```
<asp:TextBox
  BackColor = "Orange"
  ForeColor = "DarkGreen"
  Runat = "server" />
<asp:Button
  BackColor = "Orange"
  ForeColor = "DarkGreen"
  Font-Bold = "true"
  Runat= "server" />
```

文件中将两种控件的背景色都定义成 Orange，前景色定义成 DarkGreen。将 Button 控件的字体定义成粗体。

值得注意的是，有的控件(如 LoginView、User Control 等)不能用 skin 文件来定义外貌。能够定义的控件也只能定义它们的外貌属性，其他行为属性(如 AutoPostBack 属性等)不能在这里定义。

(5)　在同一个主题目录下，不管定义了多少个皮肤文件，系统都会自动将它们合并成为一个文件。

(6)　网站中凡需要使用主题文件的网页，都需要在网页的定义语句中增加“Theme = [主题目录]”的属性。例如：

```
<%@ Page Theme="Themes1"… %>
```

在设计阶段，看不出皮肤文件定义的作用，只有当程序运行时，在浏览器中才能够看到控件外貌的变化。图 19.2 是 TextBox 和 Button 控件在上述定义中显示的界面。

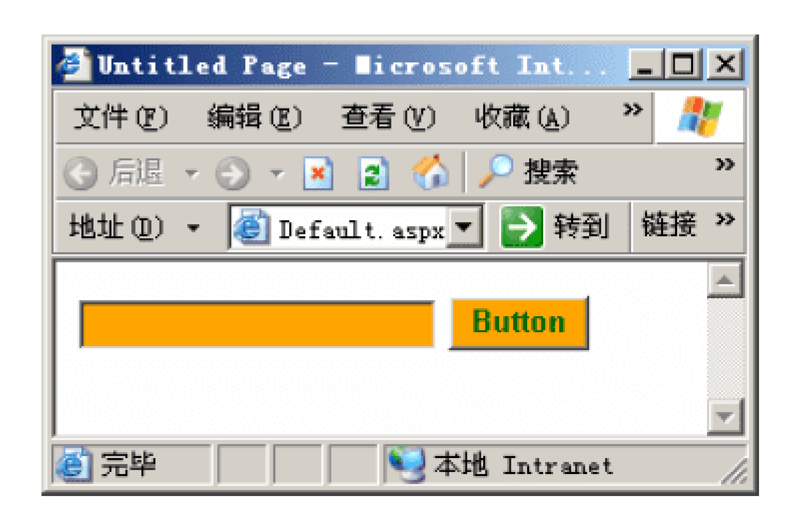


图 19.2　用皮肤文件定义控件的外貌

19.1.3　对同一控件多种定义的方法

有时需要对同一种控件定义多种显示风格，此时可以在皮肤文件中，在控件显示的定义中用 SkinID 属性来区别。例如在 TextBox.skin 文件中对 TextBox 的显示定义了三种显

示风格。

```
<asp:TextBox
     BackColor="Green"
     Runat="Server" />

<asp:TextBox
     SkinID="BlueTextBox"
     BackColor="Blue"
     Runat="Server" />

<asp:TextBox
     SkinID="RedTextBox"
     BackColor="Red"
     Runat="Server" />
```

其中第一个定义为默认的定义，中间不包括 SkinID。该定义将作用于所有不注明 SkinID 的 TextBox 控件。第二和第三个定义中都包括 SkinID 属性，这些定义只能作用于 SkinID 相同的 TextBox 控件。

在网页中为了使用主题，应该做出相应的定义。例如：

```
<%@ Page Theme="Themes1" %>
<html>
<head runat="server">
   <title>用皮肤文件定义 TextBox 的外貌</title>
</head>
<body>
   <form id="form1" runat="server">

   <asp:TextBox
        id="TextBox1"
        Runat="Server" />

   <br />

   <asp:TextBox
        id="TextBox2"
        SkinID="BlueTextBox"
        Runat="Server" />

   <br />

   <asp:TextBox
        id="TextBox3"
        SkinID="RedTextBox"
        Runat="Server" />

   </form>
</body>
</html>
```

程序运行中三个 TextBox 控件分别显示不同的风格，情况如图 19.3 所示。

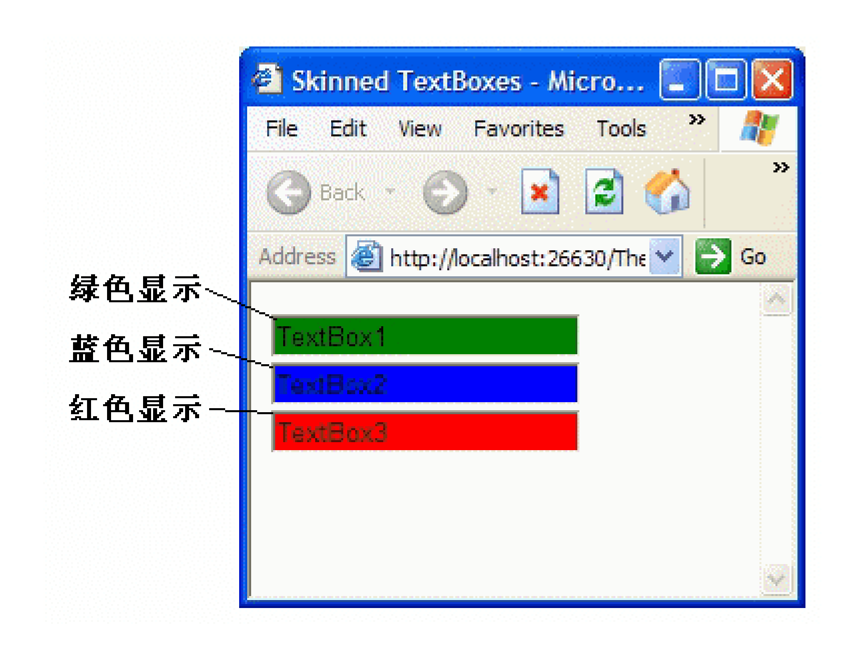


图 19.3　不同定义下的三个 TextBox 控件

实际上，每个常用服务器控件的属性对话框中都有一个 SkinID 属性，通过该属性的下拉列表框可以直接设定在皮肤文件中的选项。

19.1.4　应用主题的方法

主题既可以应用于单个网页，也可以应用于整个网站。当应用于单个网页时，需在网页的第一行代码中加入以下代码。

```
<%@ Page theme= "Themes1"…%>
```

当应用于整个网站时，应在网站根目录下的 Web.config 文件中进行配置。例如要将 Themes1 主题目录应用于网站的所有网页中，可以在 Web.config 文件中配置如下。

```
<configuration>
  <system.web>
    <pages theme="Themes1" />
  </system.web>
</configuration>
```

有了这个配置，就不用在每个网页中去定义主题文件了。如果在配置了网站共用主题的前提下，某张网页对主题有特殊要求时，还可以在该网页的 Page 语句中定义自己需要的主题，此时网页中的定义将覆盖 Web.config 文件中配置的主题。

19.2　用 户 控 件

19.2.1　什么是用户控件

用户控件(User Control)是一种自定义的组合控件，通常由系统提供的可视化控件组合而成。在用户控件中不仅可以定义显示界面，还可以编写事件处理代码。当多个网页中包括有部分相同的客户界面时，可以将这些相同的部分提取出来，做成用户控件。

一个网页中可以放置多个用户控件。通过使用用户控件不仅可以减少编写代码的重复劳动，还可以使得多个网页的显示风格一致。更为重要的是，一旦需要改变这些网页的显示界面时，只需要修改用户控件本身，经过编译后，所有网页中的用户控件都会自动跟随

变化。

用户控件本身就相当于一个小型的网页，同样可以为它选择单文件模式或者代码分离模式。然而用户控件与网页之间还是存在着一些区别，这些区别包括：

- 用户控件文件的扩展名为.ascx 而不是.aspx；代码的分离(隐藏)文件的扩展名是.ascx.cs 而不是.aspx.cs。
- 在用户控件中不能包含 <HTML>、<BODY> 和 <FORM> 等 HTML 的标记。
- 用户控件可以单独编译，但不能单独运行。只有将用户控件嵌入到.aspx 网页文件中时，才能和 ASP.NET 网页一起运行。

除此之外，用户控件与网页非常相似。

19.2.2 创建用户控件的方法

创建用户控件的步骤如下。

(1) 创建一个网站。

(2) 右击网站中某个目录，选择【添加新项】命令。在打开的对话框中选择【Web 用户控件】，然后确定用户控件的名称，单击【添加】按钮。

(3) 从工具箱中将控件添加到 Web 用户控件中。其中凡是希望用服务器编程方式访问的控件都必须是服务器端控件。

(4) 为各控件设置属性或编写事件代码。

(5) 给用户控件进行编译。方法是先选择用户控件名，然后选择【生成】|【生成页】命令以便完成编译工作。

下面结合一个示例来讲述创建用户控件的过程。

假定某个项目中多个网页的上方都需要放置如图 19.4 所示的显示界面。可以为这个界面创建一个用户控件。

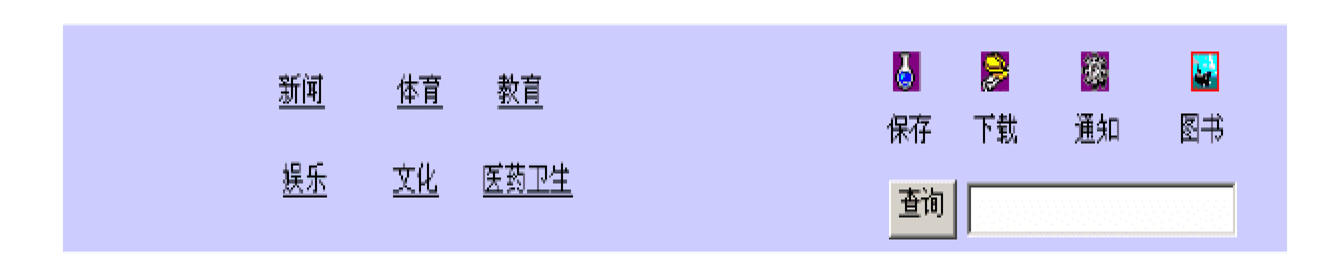


图 19.4 用户控件示例

具体操作步骤如下。

(1) 建立网站，给网站取名为 usrControl。

(2) 右击网站(应用程序)名，在弹出的菜单中选择【添加新项】命令。在打开的对话框中选择【Web 用户控件】，根据需要确定其名称，然后单击【打开】按钮，以便在设计器中打开用户控件。

(3) 从工具箱的【标准】选项卡中将 Panel 控件拖入设计窗口，设置它的属性(如底色、大小等)。然后在其中拖入其他控件，如图 19.4 所示。这些控件包括：

- 6 个 HyperLink 控件，将它们的 Text 属性分别设成“新闻”、“娱乐”等。
- 4 个 Image 以及 4 个 Label 控件，分别选择各自的图标，标上“保存”、“下载”等。

- 1 个 TextBox 文本框和 1 个【查询】按钮。

(4) 在应用程序中添加多个 Web Form(网页)，用于显示各种不同的内容。

(5) 通过 HyperLink 的 NavigateUrl 属性分别与各窗体链接。

(6) 如果需要，可双击【查询】按钮，在代码隐含文件中编写查询程序。最后选择【生成】|【生成页】命令，以编译用户控件。

19.2.3　使用用户控件

用户控件只能在同一应用程序的网页中共享。就是说，应用项目的多个网页中可以使用相同的用户控件，而每一个网页可以使用多种不同的用户控件。如果一个网页中需要使用多个用户控件时，最好先进行布局。然后再将用户控件分别拖到相应的位置。

在设计阶段，有的用户控件并不会充分展开，而是被压缩成小长方形，此时它只起占位的作用。程序运行时才会自动展开。

19.2.4　代码分析

打开 ASP.NET 的【源】视图，可以看见用户控件的相关代码如下。

```
<%@ Register TagPrefix="uc1" TagName="WebUserControl1"
    Src="WebUserControl1.ascx" %>
     <body >
       <form id="Form1" method="post" runat="server">
         <uc1:WebUserControl1 id="WebUserControl11"
              runat="server"></uc1:WebUserControl1>
     </form>
   </body>
```

其中语句

```
<%@ Register TagPrefix="uc1" TagName="WebUserControl1"
            Src="WebUserControl1.ascx" %>
```

代表用户控件已经在.aspx 中注册。语句中各个标记的含义如下。

- TagPrefix：代表用户控件的命名空间(这里是 ucl)，它是用户控件名称的前缀。如果在一个.aspx 网页中使用了多个用户控件，而且在不同的用户控件中出现了控件重名的现象时，命名空间是用来区别它们的标志。
- TagName：是用户控件的名称，它与命名空间一起来唯一标识用户控件，如代码中的 uc1:WebUserControl1。
- Src：用来指明用户控件的虚拟路径。

19.2.5　将 Web 窗体页转换为用户控件

将 Web 窗体页转换为用户控件的目的，是为了将该窗体转换成为可重用的控件。由于两者原本采用的技术就非常相似，所以只需要做一些较小的改动即能将 Web 窗体改变成为用户控件。

由于用户控件必须被嵌套于网页中运行，因此在用户控件中不能包括<html>、<body>

和<form>等 HTML 标记，否则将会产生代码重复。转换中必须移除窗体页中的这些标记。除此之外，还必须在包括用户控件的 Web 窗体页中将 ASP.NET 指令类型从@ Page 更改为@ Control。转换的具体步骤如下。

(1) 在代码(隐藏)文件中将类的基类从 Page 更改为 UserControl 类。这表明用户控件类是从 UserControl 类继承的。

例如：在 Web 窗体页中，类 Welcome 从 Page 类继承。语句如下。

```
public class welcome : System.Web.UI.Page{…}
```

现在改为从 UserControl 类继承。语句如下。

```
public class welcome : System.Web.UI.UserControl{…}
```

(2) 在.aspx 文件中删除所有<html>、<head>、<body> 和 <form>等标记。

(3) 将 ASP.NET 指令类型从@ Page 更改为 @ Control。

(4) 更改 Codebehind 属性来引用控件的代码(隐藏)文件(ascx .cs)。

(5) 将.aspx 文件的扩展名(后缀)更改为 .ascx。

19.3 母 版 页

19.3.1 什么是母版页

母版页(Master Page)是一个以.master 作为后缀的文件。一些需要网站共享的内容，如网站商标(Logo)、主菜单(Mainmenu)、常用的友好链接等都可以放置在母版页中。在母版页中既可以放进各种类型的控件，也可以编写事件处理代码，同时还要给各网页窗体留出一处或多处“自由空间”。

母版页与用户控件之间的最大区别在于，用户控件是基于局部的界面设计，而母版页是基于全局性的界面设计。用户控件只能在某些局部上使各网页的显示取得一致，母版页却可以使得在整体外观上取得一致。用户控件通常被嵌入到母版页中一起使用。

一个网站可以设置多种类型的母版页，以满足不同显示风格的需要。

19.3.2 创建母版页的方法

下面通过示例来讲述创建母版页的方法。

首先创建一个新网站，然后右击应用程序名，在弹出的菜单中选择【添加新项】命令。在弹出的对话框中选择【母版页】，并使用 MasterPage1.master 默认名(可改名，但后缀不能改)。此时在界面中将出现一个 ContentPlaceHolder 方形窗口，这个方形窗口是配置网页的地方。可以先对网页进行布局，然后再将这个窗口移动到合适的地方。下面举例说明具体步骤。

最好采用 div+CSS 并结合表格的方法进行布局，然后将 ContentPlaceHolder 拖放到相应的<div>…</div>中。

由此形成的母版页如图 19.5 所示。



图 19.5　母版页示例

19.3.3　在母版页中放入新网页的方法

可以直接在母版页中生成新网页，也可以在建立新网页过程中选择母版页。

1. 直接从母版页中生成新网页

直接从母版页中生成新网页的步骤如下。

(1)　打开母版页。

(2)　右击 ContentPlaceholder 控件，在弹出的菜单中选择【添加内容页】命令，以确定内含的新网页。

(3)　右击新网页，在弹出的菜单中选择【编辑主表】命令，然后在网页中增添新控件。

此时新网页将被嵌入到母版中，与母版页形成一个网页文件，网页的名字即新网页的名字。

2. 在创建新网页中选择母版页

在创建新网页中选择母版页的方法是：在网站中创建一新网页。此时，在网页名的右方提供了两项选择，可以从中选择一项或两项，或者两项都不选择。两种选择项的含义如下。

- 将代码放在单独的文件中：代表采用代码分离方式。
- 选择母版页：代表将新网页嵌入到母版页中。

如果两项都不选择时，系统将创建一个单文件模式的独立网页，此网页将独立于母版页。

如果选择了第二项，将弹出一个文件列表，提供一个或多个“母版页”文件以供选择。当选择其中之一后，新网页就会嵌入到指定的母版页中。

母版页与新网页将构成一个整体，成为一个新的网页，新网页仍使用新产生的网页名。在母版页中将包括多个网页的共性部分，被嵌入的网页中包含的是网页的个性部分。这种关系可以用一个简单的表达式来表示。

```
default.aspx = Master Page + default.aspx
```

网页与母版页的关系如图 19.6 所示。

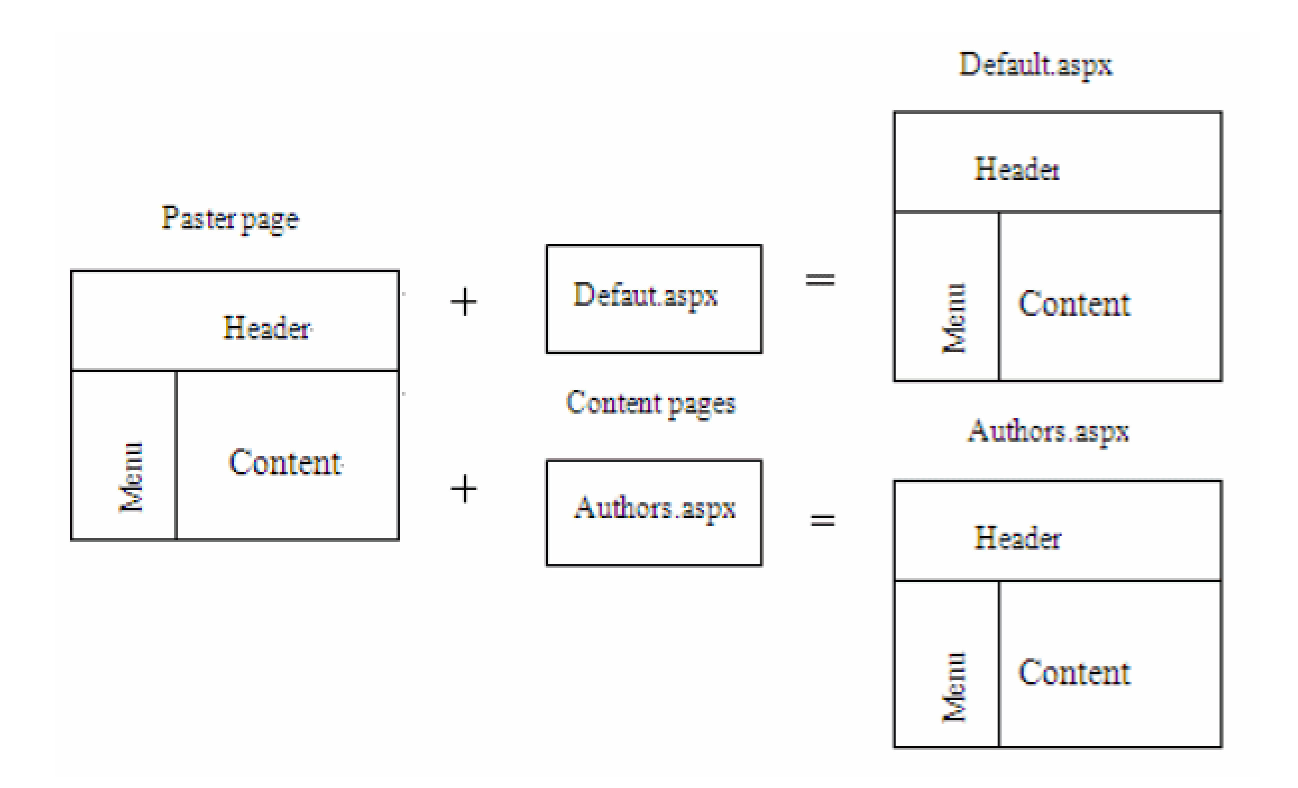


图 19.6　网页与母版页的关系

19.3.4　将已建成的网页放入母版页中

为了将已经建成的网页嵌入母版页中，需要在已经建成的网页中用手工方法增加或改写一些代码。

(1) 打开已建成的网页，打开它的代码界面，在页面指示语句中增加与母版页的联系。为此需增加以下属性，其中“~/MasterPage.master”代表母版页名。

```
<%@Page Language="C#" MasterPageFile=" ~/MasterPage.master"
AutoEventWireup="…">
```

(2) 由于在母版页中已经包含有 HTML、Head、Body、Form 等标记，因此在网页中要删除所有这些标记，以避免重复。同样，类似于<H1></H1>的标记也要删除(div 标记不要删除)。

(3) 在剩下内容的前后两端加上 Content 标记，并增加 Content 的 ID 属性、Runat 属性以及 ContentPlaceHolderID 属性，后者的值(这里是 ContentPlaceHolder1)应该与母版页中的网页容器相同。修改后的语句结构如下。

```
<asp:Content ID="bodyContent" ContentPlaceHolderID="ContentPlaceHolder1"
Runat=Server>
 <div>
   …

</div>
   </asp:Content>
```

就是说修改后的代码中除页面指示语句以外，所有语句都应放置在<asp:Content…>与</asp:Content>之间。

ASP.NET 3.5 提供了母版页之间的嵌套技术，即基于一个母版页来创建另一个新母版页。对于某些内容只需要在部分网页中共享时，可以将这部分内容嵌在主母版中，为部分网页服务。

母版页的嵌套技术并不常用，设计起来也不简单，受到的限制比一般人的想象要多，因此这里不作介绍。

19.4　小　　结

为了使网站中一批网页的显示风格保持一致，ASP.NET 提供了主题、用户控件和母版页技术。主题、用户控件和母版页虽然都是对控件显示的定义，但是它们定义的层次和影响的范围不同。

主题是利用皮肤文件对一批单个控件外貌的定义，皮肤文件必须放在主题目录之下，而主题目录又必须放在专用目录 App_Themes 之下。用户控件与母版页都是由设计者利用标准控件自行创建的组合控件，用户控件只能作用于网页的局部，而母版页是对整体布局的定义。

恰当地将三者结合，就可以使网站的多个网页之间，从单个控件到局部，再到整体布局方面在显示风格上取得一致。

19.5　习　　题

1. 填空题

(1) 皮肤文件是以.skin 为后缀的文件，用来定义________________的样式。

(2) 下面是一段皮肤文件中的定义。

```
<asp:TextBox
  BackColor = "Orange"
  ForeColor = "DarkGreen"
  Runat = "server" />
```

代码将__________服务器控件的底色定义为_________色，将控件中的字符定义为_________色。

(3) 下面是 ASPX 网页中的一段代码。

```
<%@ Register TagPrefix="uc1" TagName="WebUserControl1"
                Src="WebUserControl1.ascx" %>
```

其中带下划线的字符串代表_______________。

2. 选择题

(1) 当一种控件有多种定义时，用_______属性来区别它们的定义。

A. ID　　B. Color　　C. BackColor　　D. SkinID

(2) 用户控件是后缀为________的文件。

A. .master　　B. .asax　　C. .aspx　　D. .ascx

(3) 母版页是后缀为_________的文件。

A. .master　　B. .asax　　C. .aspx　　D. .ascx

(4) 下面是 ASPX 网页中的一段代码。

```
<%@Page Language="C#" MasterPageFile=" ~/MasterPage.master"
        AutoEventWireup="…">
```

其中带下划线的部分代表__________。

A. 母版页的路径　　B. 用户控件的名字

C. 用户控件的路径　　D. 母版页的名字

3. 判断题

(1) 利用主题可以为一批服务器控件定义样式。 (　　)

(2) 主题目录必须放在专用目录 App_Themes 的下面，皮肤文件必须放在主题目录下面。 (　　)

(3) 用户控件是一种自定义的组合控件。 (　　)

(4) 用户控件不能在同一应用程序的不同网页间重用。 (　　)

(5) 使用母版页是为了多个网页在全局的样式上保持一致。 (　　)

4. 简答题

(1) 为了保持多个网页显示风格一致，ASP.NET 3.5 使用了哪些技术，每种技术是如何发挥作用的？

(2) 简述将 ASPX 网页转换成用户控件的方法。

(3) 简述将已经创建的 ASPX 网页放进母版页的方法。

5. 操作题

将主题、用户控件及母版页技术相结合创建风格一致的多个网页。

第 20 章　网 站 导 航

一位旅客到一个陌生的地方去旅游，如果有一位好的向导，就能够顺利地看到希望观赏的景观。同样，人们到一个大型网站中去浏览，如果有一个良好的导航工具，就可以快速而方便地找到感兴趣的信息。

如何设计导航工具呢？若用传统方法来设计，往往需要编写数十条代码。现在 ASP.NET 3.5 提供了层次控件和网站地图，用它们来作向导时，可以随时查看到自己所处的位置、各网页之间的关系，因而可以给客户进一步浏览提供参考。

本章先介绍单个控件，然后介绍联合使用的方法。具体问题包括:

- TreeView 控件。
- 站点地图文件。
- 将 TreeView 结合站点地图进行导航。
- 利用动态菜单进行导航。
- 使用 SiteMapPath 控件。

20.1　TreeView 控件

20.1.1　概述

开发 Windows 桌面系统时，程序员会经常用到 TreeView 层次控件，然而在网站中，TreeView 控件的使用却并不普遍。为什么呢？这是因为在网站中，要实现像桌面系统那样的功能比较困难，传统的方法是在客户端结合“层”编写大量的 DHTML 脚本。这种方法已经被证明是一项比较艰难的工作，设计中必须考虑各种类型的 Web 浏览器以及浏览器的不同版本，设计完成以后测试也比较困难。现在使用 ASP.NET 3.5，所有这些复杂性都不复存在，系统将很多方法的复杂性封装到了控件内部，只留下了少量接口提供给设计者。

TreeView 控件可以用来表示层次型的数据。网站中各个网页之间是一种层次关系，因此很适合用 TreeView 控件来描述。不过应该明确，这里指的网站结构是指网站的逻辑结构而不是物理结构，逻辑结构与物理结构之间并不一定完全对应。这个特点增加了浏览程序设计的灵活性。

在 ASP.NET 3.5 中，既可以直接在 TreeView 控件中创建网站的逻辑结构，也可以先建立站点地图，然后再映射到 TreeView 控件中。下面将分别讲述这两种创建的方法。

20.1.2　选择 TreeView 控件的视图

系统给 TreeView 控件提供了近 20 种视图，其中包括几种比较著名的界面，如 Custom、MSDN、XPFileExplorer 和 Windows_Help。设计时可以根据需要和个人爱好选择

其中之一。选择方法如下。

双击工具箱中的 TreeView 控件图标，将 TreeView 控件放到窗体页中。在控件出现的同时，将弹出一个【TreeView 任务】对话框，在对话框中选择【自动套用格式】项以打开一个新的对话框。对话框的左边列出了不同视图的名字，单击其中之一时，右边将显示该视图的显示界面，如图 20.1 所示。

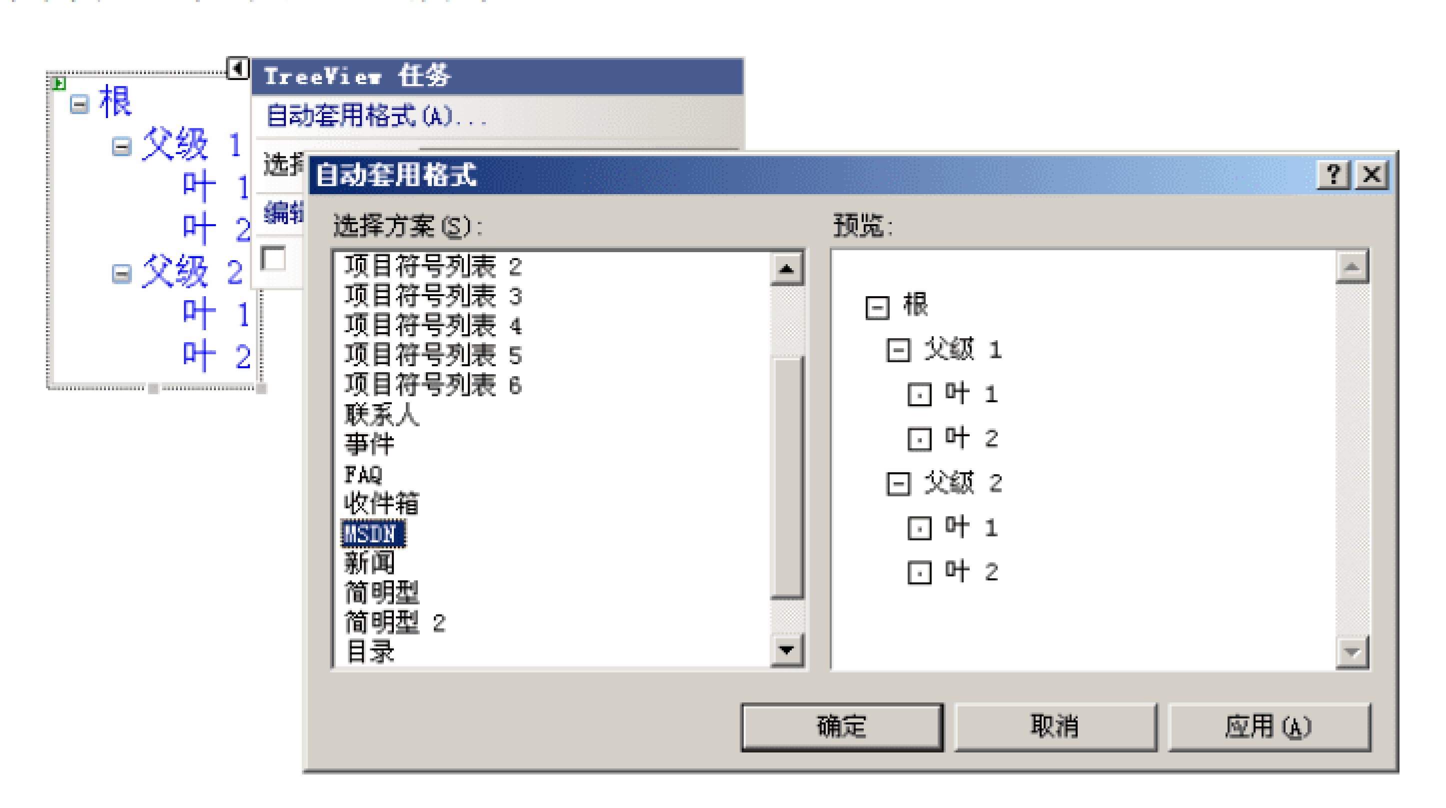


图 20.1　给 TreeView 套用格式

选择其中之一后，单击【确定】按钮，即确定了 TreeView 的显示界面。

20.1.3　编辑节点

1. 增/删节点

在【TreeView 任务】对话框中选择【编辑节点】项，在【TreeView 节点编辑器】中利用几个增、删按钮增减网页名，然后将各网页通过 NavigateUrl 属性与实际网页相连，如图 20.2 所示。

2. 给节点属性赋值

TreeView 控件的属性很多，常用的属性如下。

- EnableClientScript 属性：这是一个重要的属性，默认为 true。这表明允许用客户端脚本来处理展开和折叠节点的事件，从而避免在展开和折叠节点时与服务器之间进行代价昂贵的信息往返。如果将该属性设置为 false，则每当客户单击树中的节点时，都需要向服务器进行信息发送和处理。
- ShowLines 属性：默认情况下各节点之间没有用线条连接。如果希望在节点之间用线条连接时，可以将 ShowLines 属性设为 true。
- Show CheckBoxes 属性：是否在节点上显示复选框。默认为 None，代表不显示复选框。如果需要显示复选框，在该属性的下拉列表中还包括了多种选择。
    - Root：在根节点上加复选框。
    - Parent：在父节点上加复选框。
    - Leaf：在叶子节点上加复选框。
    - All：在所有节点上加复选框。

- ExpendDapth 属性：初始情况下节点显示的深度。默认是 FullyExpand，代表将显示全部深度上的节点。可以将该属性设置成一个数字(例如 2)，以确定初始条件下显示的深度。

图 20.3 是将 TreeView 控件的 ShowLines 属性设为 true，并且将 Show CheckBoxes 属性设为 All 后显示的界面。



图 20.2 给 TreeView 编辑节点

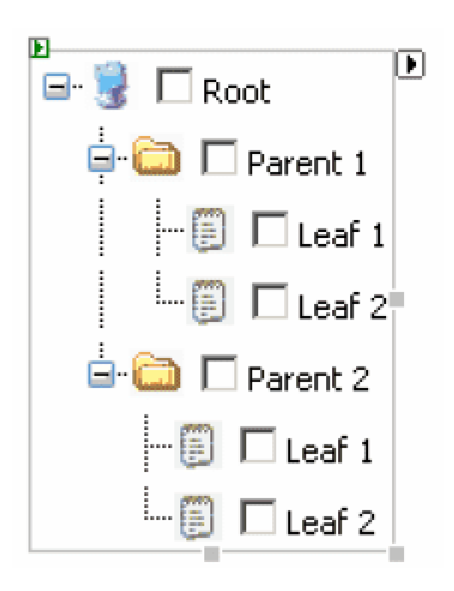


图 20.3 给 TreeView 增加连线并在所有节点上添加复选框

20.1.4 对节点事件的处理

对 TreeView 控件的操作包括两种不同的情况：一种是展开或折叠树节点；另一种是引发其他的操作，多数情况下是打开一个新网页。为了提高运行效率，在各个节点的 SelectAction 属性中，定义了几种不同的处理方式。

- Select：引发 SelectNodeChanged 事件，在服务器端处理。
- Expand：引发 TreeNodeExpanded 事件以展开树节点，在浏览器端处理，不发送给服务器。
- SelectExpand：既引发 TreeNodeExpanded 事件，又引发 SelectNodeChanged 事件。
- None：不引发任何事件。

若选择 Select(这是默认方式)或者 SelectExpand 方式还需要设置以下两个属性。

- NavigateUrl：被调用网页的 URL。
- Target：被打开网页放置的位置。

选项中的默认项是 Select。如果仅仅为了展开/折叠节点，选用 Expand 选项比较有利，因为该选项将自动在浏览器端执行，运行的效率较高。

例如将“网页 2_2”节点的事件设置为 Expand 时的情况如图 20.4 所示。

设置后的部分代码如下。

```
<asp:TreeView ID="TreeView1" runat="server" ImageSet="Msdn"
NodeIndent="10" ShowCheckBoxes="Leaf">
```

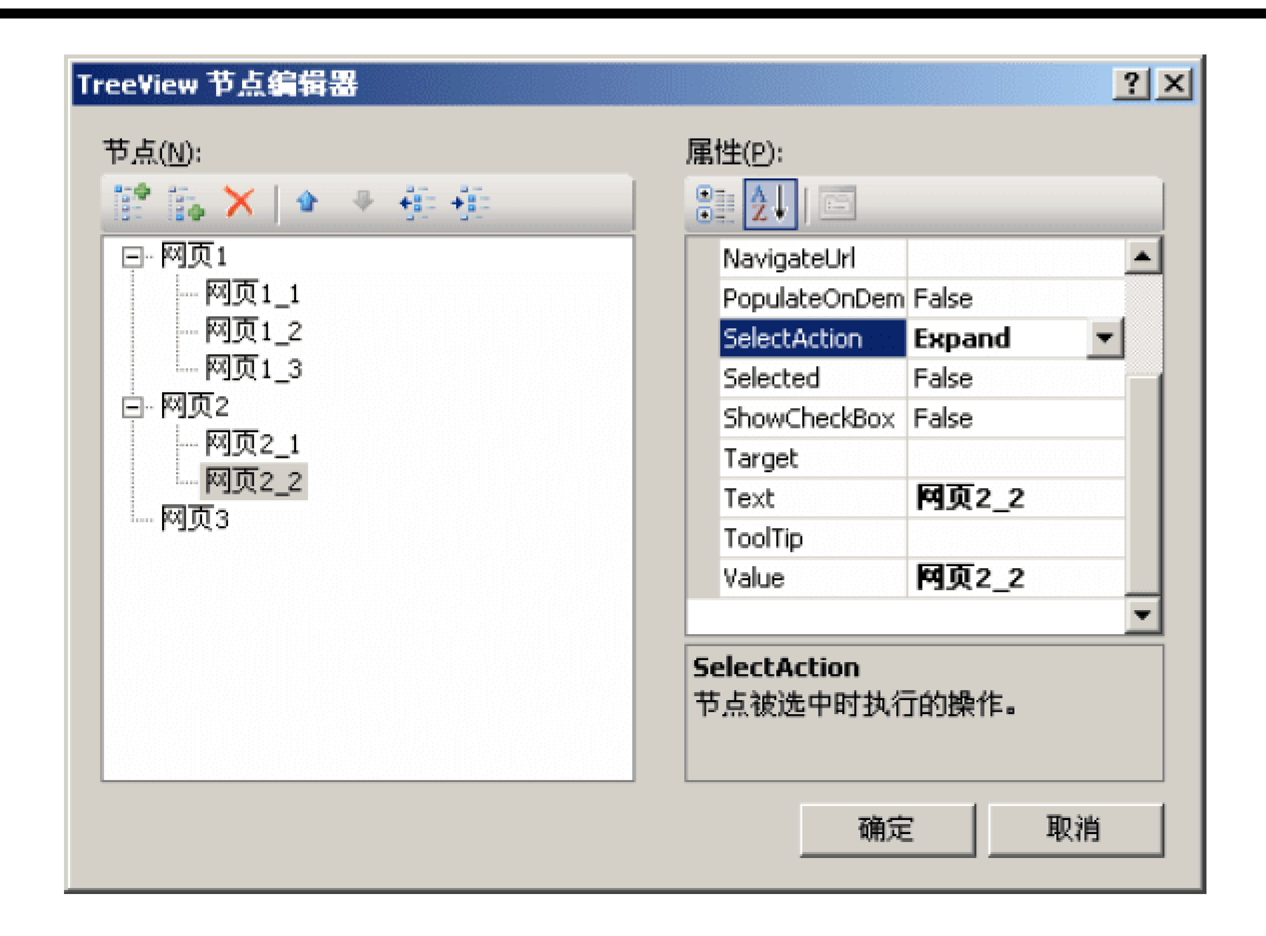


图 20.4　将某节点的 SelectAction 属性设置为 Expand

```
<Nodes>
  <asp:TreeNode Expanded="True" Text="网页 1" Value="网页 1">
     <asp:TreeNode Text="网页 1_1" Value="网页 1_1"></asp:TreeNode>
     <asp:TreeNode Text="网页 1_2" Value="网页 1_2"></asp:TreeNode>
     <asp:TreeNode Text="网页 1_3" Value="网页 1_3"></asp:TreeNode>
  </asp:TreeNode>
  <asp:TreeNode Text="网页 2" Value="网页 2">
     <asp:TreeNode Text="网页 2_1" Value="网页 2_1">
        <asp:TreeNode Text="网页 2_1_1" Value="网页 2_1_1">
          <asp:TreeNode Text="网页 2_1_1_1" Value="网页 2_1_1_1">
</asp:TreeNode>
        </asp:TreeNode>
     </asp:TreeNode>
     <asp:TreeNode Text="网页 2_2" Value="网页 2_2">
</asp:TreeNode>
  </asp:TreeNode>
  <asp:TreeNode Text="网页 3" Value="网页 3">
</asp:TreeNode>
</Nodes>
 </asp:TreeView>
```

其中，“网页 1”设置为 Expand。其他网页节点均采用默认方式，即 Select 方式。

20.2　站点地图文件

站点地图文件是一个 XML 文件，用来描述网站的逻辑结构。文件的后缀是.sitemap。有几种层次控件都可以结合站点地图所确定的逻辑关系来给网站导航。

为了创建站点地图，右击网站，在弹出的菜单中选择【添加新项】命令，然后选择【站点地图】选项，以打开站点地图文件。在站点地图文件中有多个节点，每个节点具有 3 种属性：节点标志、调用网页的 URL 以及内容提示等。具体情况如下。

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
   <siteMapNode url="" title=""  description="">
      <siteMapNode url="" title=""  description="" />
      <siteMapNode url="" title=""  description="" />
   </siteMapNode>
</siteMap>
```

其中：

- url：该节点调用网页的 URL。
- title：节点的标志。
- description：作为智能提示的字符串(IntelliSense)。

站点文件刚被打开时只是给出了一个框架，具体内容还需要用手工在“”中去填写。这个框架表明，文件包括了一个根元素（<siteMap>)，零个或多个嵌套的节点元素<siteMapNode>。

下面是一个填写后的简单示例。

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap>
  <siteMapNode title="公司介绍" description="Home" url="default.aspx" >
    <siteMapNode  title=" 计 算 机 产 品 "  description=" 本 公 司 产 品 "
url="default2.aspx">
       <siteMapNode title="软件" description="软件选择" url="default3.aspx" />
       <siteMapNode title="硬件" description="硬件选择" url="default4.aspx" />
    </siteMapNode>
    <siteMapNode title="服务" description="公司提供的服务" url="default5.aspx">
      <siteMapNode title="培训" description="培训课程"  url="default6.aspx" />
      <siteMapNode title="咨询" description="咨询的项目" url="default7.aspx" />
    </siteMapNode>
</siteMapNode>
</siteMap>
```

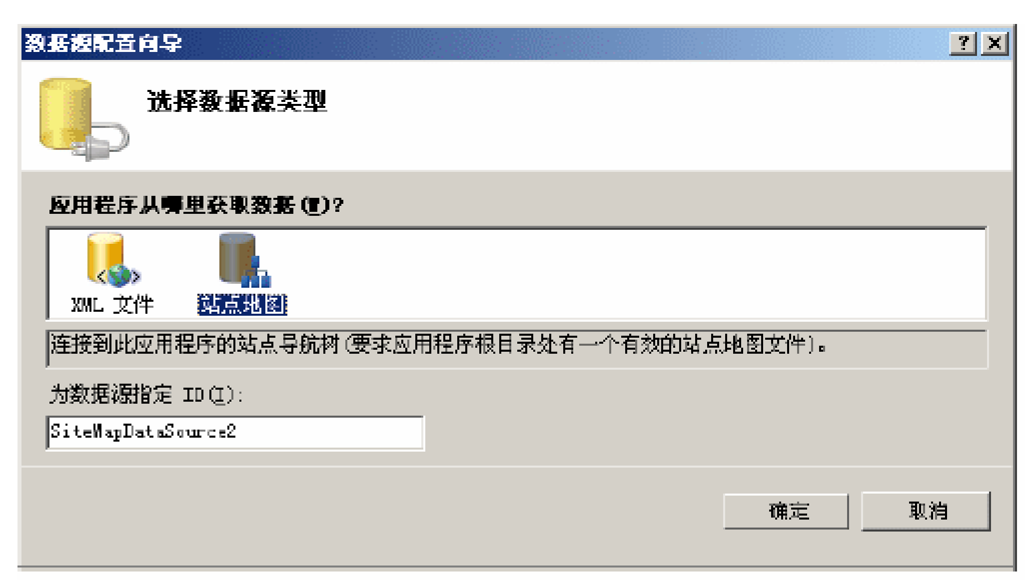
文件中“公司介绍”是最外层的节点，它的下面包括“计算机产品”和“服务”两大子节点。“计算机产品”节点下面又包括“软件”、“硬件”两个子节点。

20.3　将 TreeView 结合站点地图进行导航

Web TreeView 控件利用站点地图进行导航的步骤如下。

(1) 将 TreeView 控件置于窗体中，选择新数据源控件，将弹出如图 20.5 所示的对话框。

(2) 选择站点地图文件作为数据源，将自动产生 SiteMapDataSource 数据源控件并且建立与 Web.sitemap 文件的连接。运行程序时将出现如图 20.6 所示的界面。



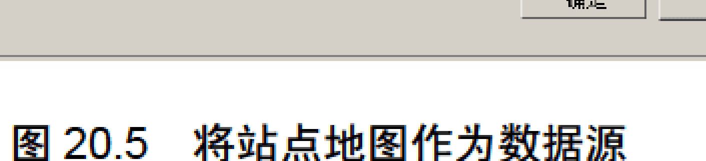


图 20.5 将站点地图作为数据源

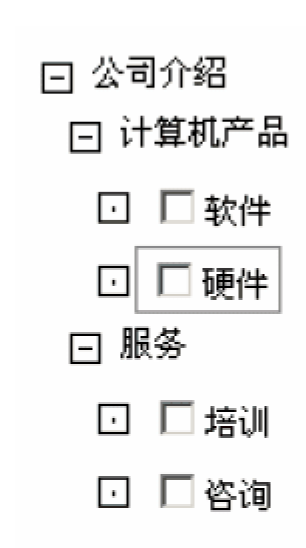


图 20.6 TreeView 的显示界面

20.4 利用动态菜单进行导航

利用动态菜单控件并结合站点地图文件同样可以给网站导航。程序运行时只要将鼠标指针移动到菜单的某个节点上时，就会自动弹出其下一层的节点，当鼠标指针离开该节点后子节点又会自动消失，整个显示的过程是动态的。

20.4.1 结合站点地图创建动态菜单

现在结合 20.2 节中使用的站点地图来创建动态菜单，其步骤如下。

(1) 从工具箱中将菜单(Menu)控件放入窗体页中。其他设置与对 TreeView 控件设置的方法基本相同。将站点地图作为菜单的数据源，此时的界面如图 20.7 所示。

(2) 和 TreeView 控件显示的方式不同，它并不是同时将菜单内容全部显示出来。只要当鼠标指针移动到某个节点时，该节点的子节点就会自动弹出，当鼠标指针移开时，弹出的界面也会自动消失。情况如图 20.8 所示。



图 20.7 将站点地图作为动态菜单的数据源

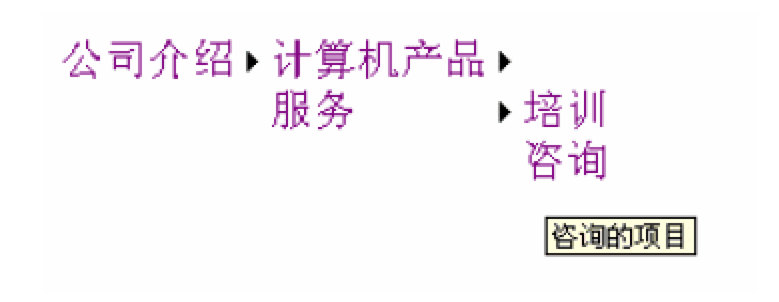


图 20.8 菜单的动态显示

20.4.2 创建主菜单

菜单(Menu)的属性 Orientation 可以用来确定菜单项的排列方式。该属性包括的方式有两种：竖向排列(Vertical)与横向排列(Horizontal)。默认时为竖向排列方式。

当改为横向排列时，菜单中的主项将排列成一行。将这种菜单放入母版页中作为主菜单可以给网站导航带来很大方便。

为了简化操作，可以直接采用手工方法来编辑主菜单的菜单项。图 20.9 就是一个手工编辑菜单项的示例。

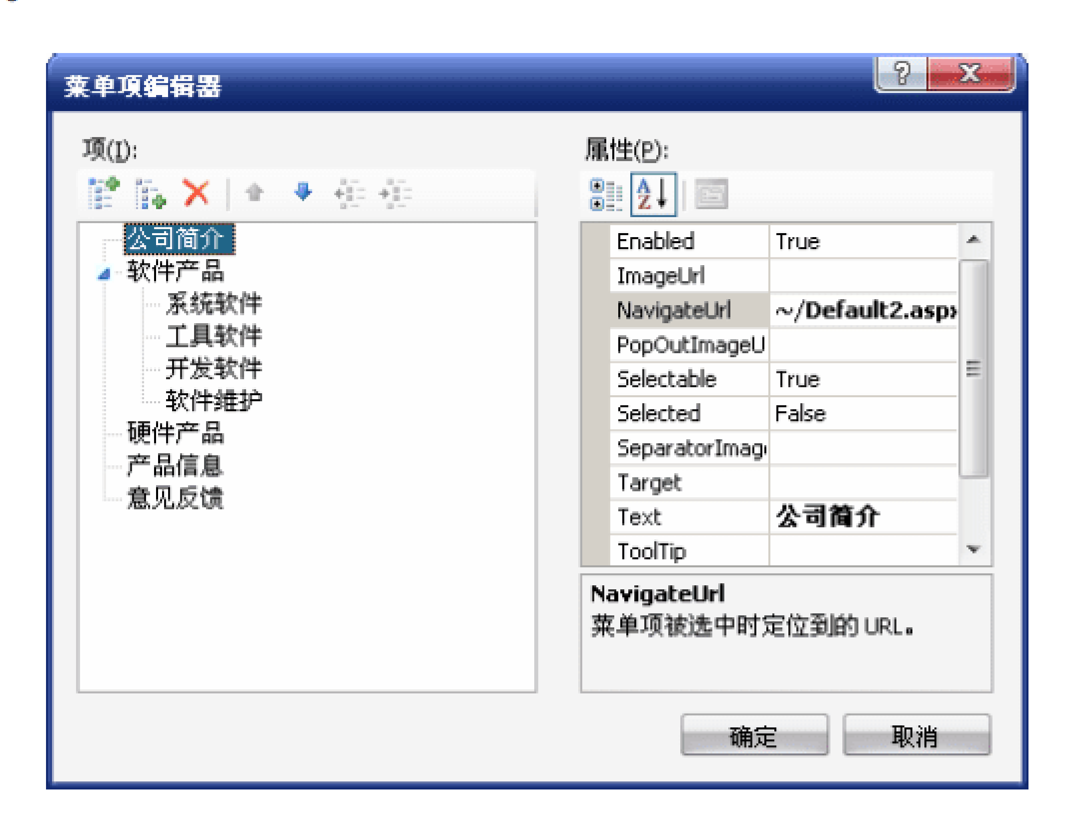


图 20.9　手工编辑菜单项示例

该菜单在网页中的显示如图 20.10 所示。

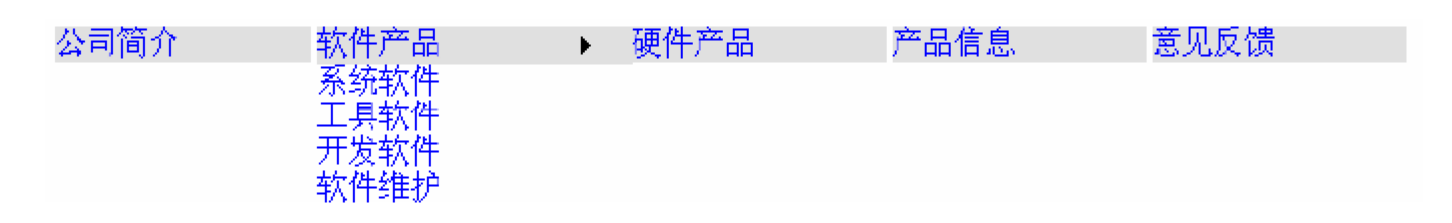


图 20.10　主菜单显示示例

20.5　使用 SiteMapPath 控件

网站路径(SiteMapPath)控件用来显示浏览者当前的位置，它必须与网站地图相结合，而且最好放在母版页中。不需要为它编写什么代码，只要应用程序中有写好的站点文件，将 SiteMapPath 控件拖入窗体时，它就会自动与站点文件结合。

网站路径控件只能显示从根节点到当前节点之间的路径，利用它只能返回到某个页面，而不能向前选择页面。其显示情况如下。

根节点 > 父节点 > 当前节点

在网站路径控件中，除有一些通用的属性以外，还有一些特殊的属性可以用来改变控件的显示界面。

- PathDirection：显示路径的方向。包括两种选择。
  - RootToCurrent：从根到当前网页。
  - CurrentToRoot：从当前网页到根。
- PathSeparator：指定网页之间的分隔符，在这里可以选择不同的分隔符号。
- RenderCurrentNodeAsLink：这是一个逻辑值(true 或 false)，确定是否使当前网页也和其他网页一样以超链接方式显示。

20.6 小　　结

网站中的文件是一种层次结构。TreeView 和 Menu 都是层次控件，因此非常适合于作导航控件。可以直接在控件中设置节点并建立它们之间的逻辑关系，然后再与网站中的实际网页连接起来。也可以先建立站点地图，设置节点以及它们之间的逻辑关系，然后以站点地图作为数据源与层次控件进行数据绑定。由于这里建立的站点是逻辑结构，不是物理结构，因此具有很大的灵活性。

SiteMapPath 控件是一个比较特殊的控件，只要网站中设置了站点地图，该控件放进网页，就会自动与站点地图连接，并显示以根节点作为起点，直到浏览者当前位置的路径。它只能显示已经浏览过的路径，而不能显示前进方向的网页。利用它返回到前面的节点时将非常方便。

20.7 习　　题

1. 填空题

(1) Web TreeView 控件的节点的 EnableClientScript 属性为 true 时，代表允许在浏览器端处理____________________事件。

(2) 当将 TreeView 的 ExpandDepth 属性设为 3 时，代表默认时显示的__________。

2. 选择题

(1) 下面是 TreeView 控件的一段代码。

```
<asp:TreeNode Text="节点 1" SelectAction="Expand">
   <asp: TreeNode Text="节点 2"…/>
   <asp: TreeNode Text="节点 3"…/>
</ asp:TreeNode>
```

表明单击“节点 1”时只作________的操作。

A. 调用网页　　　　B. 显示或隐藏本节点

C. 折叠或展开子节点　　　　D. 恢复原状

(2) 下面是网站地图中的一段代码。

```
<siteMapNode url="" title="AAA"  description="">
<siteMapNode url="" title="BBB"  description="" >
<siteMapNode url="" title="CCC"  description="" />
</siteMapNode>
<siteMapNode url="" title="DDD"  description="" />
</siteMapNode>
```

网站中 4 个节点的关系如图______所示。

(3) 下面是网站地图中的一段代码。

```
<siteMapNode url="" title="AAA"  description="">
```

```
<siteMapNode url="" title="BBB"  description="" >
<siteMapNode url="" title="CCC"  description="" />
<siteMapNode url="" title="DDD"  description="" />
</siteMapNode>
</siteMapNode>
```

网站中 4 个节点的关系如图______所示。

图A	图B	图C
⊟ AAA ⊟ BBB CCC DDD	⊟ AAA ⊟ BBB CCC DDD	⊟ AAA BBB CCC DDD

3. 判断题

(1) Web TreeView 控件只能用来描述关系型数据。 (　　)

(2) 用 Web TreeView 控件描述的是网站的物理结构。 (　　)

(3) 在 Web TreeView 控件节点的 SelectAction 属性中的 Expand 引发的事件通常在服务器中进行处理。 (　　)

(4) 利用 SiteMapPath 控件只能显示浏览的当前位置以及经过的路径。 (　　)

4. 操作题

(1) 直接在 Web TreeView 控件中创建网站的逻辑结构。

(2) 先创建网站地图，然后在母版页中利用 TreeView、动态菜单和 SiteMapPath 控件结合网站地图进行导航。

(3) 在母版页中增加一主菜单以便进行网站导航。

第 21 章　基于角色的安全技术

因特网(Internet)虽然是一个面向全球的开放型网络系统，然而其中有些网页并不是对所有客户都无条件开放的。例如：

- 一些用于公司内部管理的网页只对公司内部的人员开放。
- 有些网站设立的收费项目，只对那些进行了注册并交纳了费用的客户开放。
- 有些商业网站实行“会员制”，只有经过注册的会员，才有权参加某些商业交易活动。
- 一些远程教育网站允许学生查阅自己的成绩但不允许修改成绩。

类似的情况还可以列出很多，这些情况给网站的设计提出了新的要求：为了网站的合法权益和网络安全，必须对一些特定的网页实施保护；当客户进入时要进行身份认证，并在认证的基础上分配资源。

基于角色的安全技术目前已经成为大多数网站必备的功能，然而设计这项功能并不简单，若使用传统的方法，需要使用十几个标准控件，编写上百行代码，并经过反复的调试才能完成。现在 ASP.NET 2.0 及 3.5 与 IIS 服务器相结合，在框架的支持下，对传统的方法做了很大的改进。系统提供了强大的工具和若干组合控件。利用这些工具可以采用简易的方法，快速开发出功能完备的基于角色的安全系统。

本章将讲述以下几方面的内容：

- 基于角色的安全技术的特点。
- ASP.NET 3.5 基于角色的安全技术的特点。
- 基于角色的安全技术的准备工作。
- 利用控件创建安全网页。
- 直接调用 Membership API 方法。

21.1　基于角色的安全技术特点

21.1.1　网站中可以包括多个入口

在桌面系统中，由于入口点比较集中，因此为了保护某些窗体，主要方法就是先给客户建立注册表，表中载入客户的相关信息，然后在窗体的入口处设置登录界面来检查来访的客户，看看该客户是否在注册表中注册，如果已经注册再看看属于什么身份，然后根据其身份跳转到相应的入口。

然而网站与此不同，它是一个开放型的系统，很多网页都有自己的 URL，客户通过 URL 可以直接访问这些网页，因此网站的入口点很多。这种情况下，即使设置了登录网页，客户还可以绕过它直接进入其他网页。因此，为了保护网页需要给这些网页实施保护，或者先将它们集中到某些子目录下，再给这些子目录实施保护。

21.1.2　基于角色的安全技术是有层次的

基于角色的安全技术是有层次的，应该先将客户划分成登录客户和非登录客户，再将登录客户区分成不同的“角色”，并为这些“角色”赋予不同的访问权限。这些权限有的还需要不定时地进行调整。以交费网页为例，客户交费以前和交费以后的访问权限是不同的。

在传统的基于角色的安全技术中，首先要给被保护的网页设置保护措施(如用 Session 对象进行保护等，参见第 9 章)，并且给客户建立注册表，在注册表中注明客户的姓名、口令以及分配的角色等，如表 21.1 所示。

表 21.1　注册表

编号(BH)	姓名(Name)	密码(Pass)	角色(Role)
1	刘历丽	111111	
2	王宏贵	222222	管理员
3	李大有	333333	总经理
4	成流去	444444	
5	丁一	4433	管理员
6	李仁爱	555666	

注意：角色为空时代表一般成员。

客户的认证实质上是一个查询过程。当客户打开登录页面时，先要求客户输入自己的姓名和密码，然后到注册表中去查询。如果在表中找到了可以匹配的记录时，说明该客户已经进行了注册，然后取出客户对应的角色字段，根据分配给该角色的权限让客户转入到相应的网页。

21.2　ASP.NET 3.5 基于角色的安全技术特点

在 ASP.NET 3.5 中，为了进行客户管理并保证网站安全，系统提供了完善的服务。包括提供了一个网站管理工具和 7 个组合控件，自动建立了一个专用数据库(取名为 ASPNETDB.MDF)和若干专用数据表(存储客户注册信息、角色信息以及个性化服务的信息等)。不仅如此，系统还将存入信息、查询注册表等项工作全面地实现了自动化。在系统的支持下，设计者只需做一些简单的设置，就能设计出功能比较完备的基于角色的安全的网站。

在基于角色的安全方面主要包括两方面的工作。

(1) 客户认证方面：创建新客户；登录客户；客户修改密码；恢复客户密码；显示状态和其他有关信息。

(2) 权限管理方面：给客户划分角色；给角色分配资源。

21.3 基于角色安全技术的准备工作

21.3.1 组织好站点中的文件

为了集中入口点，最好建立若干子目录，并且将安全等级相同的网页放在同一个子目录下，然后在各个子目录下设置 Web.config 文件，利用该文件配置安全策略。这就好比一个单位中有一些保密文件，可以先将它们放到专门的办公楼里，并在办公楼的入口处设立保卫人员以检查来访的客户。Web.config 文件就相当于各个办公楼的保卫人员。图 21.1 是一个目录分布的示意图。



图 21.1　Web.config 文件与目录的关系

其中从上而下：第一个 Web.config 文件用来保护 admin 目录下的文件；第二个 Web.config 文件用来保护 manager 目录下的文件；第三个 Web.config 文件用来保护网站根目录下的文件。

Web.config 文件之间的定义有继承关系，若子目录的定义不同于父目录时，子目录下的文件按照子目录的定义执行。

21.3.2 利用网站管理工具进行安全配置

利用系统提供的“ASP.NET 网站管理工具”来定义角色，增添客户，制定访问规则，确定信息存储位置等。

1. 安全配置的步骤

选择主菜单中的【网站】|【ASP.NET 配置】命令，以打开网站管理工具，如图 21.2 所示。

选择其中的【安全】项(或者【安全】选项卡)，即可以打开安全设置界面，如图 21.3 所示。



图 21.2　网站管理工具的整体界面



图 21.3　安全设置界面

网站的安全设置包括增加“客户”(User)、定义“角色”(Role)、指定“访问规则”(Access Rule)三大部分。可以利用下面的三个方框分别进行设置，也可以利用系统提供的安全设置向导的智能提示逐项进行设置。下面重点介绍使用安全设置向导来设置的方法。

(1) 单击【使用安全设置向导按部就班地配置安全性】链接，在出现的对话框中单击【下一步】按钮。

(2) 在【选择访问方法】界面中提供了 Internet 和 Intranet 两种选择，这里选择【通过 internet】项，单击【下一步】按钮。

(3) 在【高级提供程序设置】对话框中提示：若要更改应用程序的数据存储区，请退出安全向导，选择【提供程序配置】选项卡。使用【提供程序配置】选项卡可以配置网站管理数据的存储方式。这里不改变数据存储区，即仍然使用默认提供的 SQL Server 数据库存储数据。因此只要单击【下一步】按钮即可。

(4) 在步骤(1)中，系统会询问是否创建基于角色的应用，如果应用中需要用到角色，就必须选中【为此网站启用角色】复选框，单击【下一步】按钮，打开如图 21.4 所示的界面。

本界面中表示已经创建了两个角色(如果第一次创建，将不含任何角色)，如果需要增加新角色时，先在【新角色名称】文本框中输入新角色名，然后单击【添加角色】按钮即可。如果需要删除某种角色时，只要单击该角色名右方的【删除】即可。处理完成以后单击【下一步】按钮。

(5) 输入客户名、密码、E-mail 等信息。具体做法将在 21.3.3 节中讲述。单击【下一步】按钮。

(6) 弹出的界面如图 21.5 所示。



图 21.4　增加新角色界面



图 21.5　设置访问规则界面

图中的左边列出的是网站的目录，中间的【角色】下拉列表框中将列出角色名(下面为客户名)，右方为访问权限(允许或拒绝)。将三者结合起来确定某角色(或客户)对某目录的访问权限。操作过程是：先确定角色(或客户)，再确定权限，再选择目录，最后单击【添加此规则】按钮。

2. 对安全配置的调整

如果需要修改或重新调整安全配置，可以在安全配置的初始界面中，选择下面三个选项之一。

- 客户：用来增添、编辑或删除客户，还可以给客户分配角色。
- 角色：用来增添、编辑或删除角色。
- 访问规则：用来增加、编辑或删除访问规则。

为了给客户分配角色，回到配置工具中的【安全】选项卡，单击下面【客户】中的【管理客户】项，打开如图 21.6 所示的界面。



图 21.6　给客户分配角色

分配客户的角色需逐个进行。单击某客户右边的【编辑角色】按钮时，将弹出已经存在的全部角色，单击复选框选中就代表分配了角色。一个客户可以分配 0 个或者多个角色。

检查访问权限的分配，回到配置工具中的【安全】选项卡，单击下面【访问规则】中的【管理访问规则】项，打开如图 21.7 所示的界面。



图 21.7　检查或调整权限的分配

单击左边的某个目录，然后查看右边分配的情况。图 21.7 中的结果是对于 manager 目录来说，允许 admin 与 manager 角色访问，但拒绝 sales 角色访问。如果这里显示的规则不符合需要，单击下面的【添加新访问规则】后进行调整。

右边有【上移】和【下移】两个按钮，可以用来调整规则的顺序。顺序在这里是非常重要的。因为系统总是从上到下逐个匹配规则的，如果这些规定中存在着矛盾，只有第一个被匹配的规则有效。

3. 安全配置的结果

安全配置产生了两个重要的结果。

(1) 在各个目录下分别产生了 web.config 配置文件。该文件载入了对该目录的访问权限。例如 Admin 目录的 web.config 文件如下。

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
     <authorization>
        <allow roles="admin" />
        <deny roles="manager" />
        <deny roles="sales" />
        <deny users="*" />
        <deny users="?" />
     </authorization>
   </system.web>
</configuration>
```

代码中的 allow 代表“允许”，deny 代表拒绝，roles 代表角色，users 代表客户，“*”代表所有客户，“?”代表匿名客户。这段代码的作用是，此目录下的文件只允许 admin 角色访问，拒绝 manager 及 sales 角色以及所有其他客户、匿名客户访问。在这里代码的顺序非常重要，因为系统总是按照从前向后逐条匹配的办法，并执行最先的匹配者。

利用这一特点，能够用非常简单的方法来调整客户的访问权限。例如在收费项目中，某客户的交费到期，只要将该客户的名字或角色移到“<deny users="*" />”的后面即可。一旦该客户补交了费用时，再将它移到“<deny users="*" />”的前面来，其中不需要编写任何代码。

(2) 在 App_Data 目录下出现了一个专用的 SQL Server 数据库(名为 ASPNETDB.MDF)。数据库中包括用于客户管理的若干专用数据表，这些数据表将自动记录登录客户、角色以及它们的相关数据。

4. 集中编写保护代码的方法

程序设计者也可以在根目录下的 Web.config 文件中直接编写代码，来保护各个子目录下的文件。编写的代码中要使用<location>标签的 path 属性来指定保护的范围。现举例说明如下。

假定某网站的目录结构如图 21.8 所示。

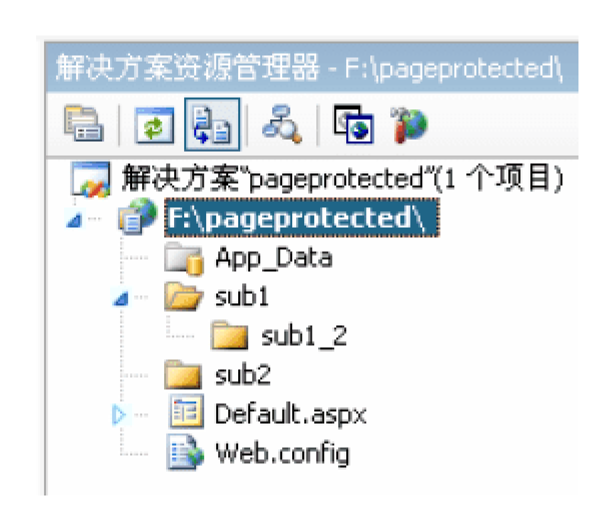


图 21.8　某网站的目录结构

打开网站根目录下的 Web.config 文件。在 <system.web>…</system.web> 下面增添各子目录的保护设置。一般情况下，根目录下的文件向所有客户开放，不必设保护，而子目录则根据需要设置保护。子目录的层次越深，通常保护也愈加严格。代码如下。

```
<configuration>
   <system.web>
     …
   <system.web>
       <location path="sub1">
         <system.web>
                <authorization>
            <allow roles="admin" />
            <allow roles="manager" />
            <deny users="*" />
          </authorization>
         </system.web>
       </location>
       <location path="sub1/sub1_2">
         <system.web>
          <authorization>
            <allow roles="admin" />
            <deny users="*" />
          </authorization>
        </system.web>
      </location>
      <location path="sub2">
        <system.web>
         <authorization>
           <allow roles="admin" />
           <allow roles="manager" />
           <deny users="*" />
         </authorization>
       </system.web>
      </location>
```

以上代码表明，对于子目录 sub1(<location path="sub1">)来说，允许 admin、manager 角色访问，而对 sub 的子目录 sub1_2(<location path="sub1/sub1_2">)来说，只允许 admin 角色访问。

当几个目录的保护设置相同时，可以将多个目录的保护代码归并到一起，再用逗号隔开各个目录名。在上面的配置中，sub1 与 sub2 两个目录就属于这种情况。归并后的代码

如下。

```
<configuration>
  <system.web>
   …
  <system.web>
     <location path="sub1,sub2">
       <system.web>
         <authorization>
          …
         </authorization>
       </system.web>
     </location>
```

21.4　利用控件创建安全网页

ASP.NET 3.5 系统提供了一组客户管理控件，这些控件中大多数都不是单一的标准控件，而是多个控件的组合。利用这些控件可以非常方便地完成客户管理和基于角色的安全策略的设计工作。这些控件包括 Login(客户登录)、CreateUserWizard(创建新用户)、LoginView(登录视图)、LoginName(登录客户名)、LoginStatus(登录状态)、ChangePassword(改变密码)、PasswordRecovery(恢复密码)

这些控件不仅定义了初步外观(可以进一步修改)，还定义了标准行为。例如有的控件可以用来创建客户的注册、登录和密码恢复界面的外形并实现其功能。也有一些控件主要用来向客户显示不同的信息。例如，利用 LoginView 控件可以定义不同的模板，将其显示给不同角色的成员等。

21.4.1　客户登录控件

1. 控件的作用

客户登录控件(Login)是基于角色的安全技术的核心控件。该控件的作用是进行客户认证，确定新到的客户是否已经登录。该控件的界面如图 21.9 所示。该控件对应的代码如下。

```
<asp:Login ID="Login1" Runat="server" />
```

开始生成的界面不一定符合需要，需要改变时，右击控件，在快捷菜单中选择【自动套用格式】命令，在弹出的对话框中可以进一步选择其他界面，并且通过属性修改界面中的显示(例如将英文显示改成汉字显示等)。

Login 控件实质上是一个“用户控件”，它不仅生成了显示界面，还定义了相应的行为。由于系统已经自动生成了数据表，而且数据表的表名、字段名以及位置都已经固定，因此只要将 Login 控件拖入到窗体中，不需要编写任何代码，也不需设置任何其他属性就可以使用。

2. 部署客户登录控件

执行 Login 控件的结果要么登录成功，要么登录失败。为了帮助客户的后续操作应该

对这两种结果都提供帮助。

当登录成功时，后续的操作如下。

(1) 转向新页面。Login 控件的 DestinationPageUrl 属性用来设置跳转的页面地址。

(2) 改变视图。利用本页面的 LoginView 控件改变视图，显示基于角色的不同界面。具体方法将在 21.4.4 节中讲述。

(3) 显示登录状态。利用 LoginStatus 控件显示登录状态，以便随时退出登录状态。

(4) 表示对登录客户的欢迎。利用 LoginName 控件编写欢迎语句。

登录失败时通常需要进行的操作如下。

(1) 提示错误信息，要求重新登录。Login 控件的 FailureText 属性用来确定登录失败时的提示文本。

(2) 创建新客户。通过 CreateUserWizard 控件创建新客户，以完成登录前的准备工作。

(3) 恢复口令。通过 PasswordRecovery 控件帮助客户恢复口令。

为此在 Login 控件中最好设置与上述控件相应的页面的连接指针。设置的方法如下。

(1) 利用属性 CreateUserText 和 CreateUserUrl 相结合指向创建的新客户界面。前者为指针的文本，后者是网页的地址。

(2) 利用属性 PasswordRecoveryText 和 PasswordRecoveryUrl 相结合指向口令代码恢复的网页。前者是指针文本，后者是网页的地址。

(3) 利用属性 HelpPageText 和 HelpPageUrl 相结合指向帮助网页。前者是指针文本，后者是网页的地址。

上述这些属性的设置以及在控件中的显示如图 21.10 所示。

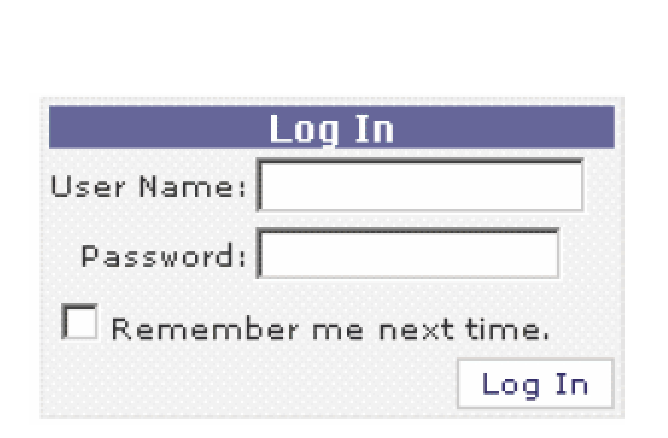


图 21.9 登录控件界面

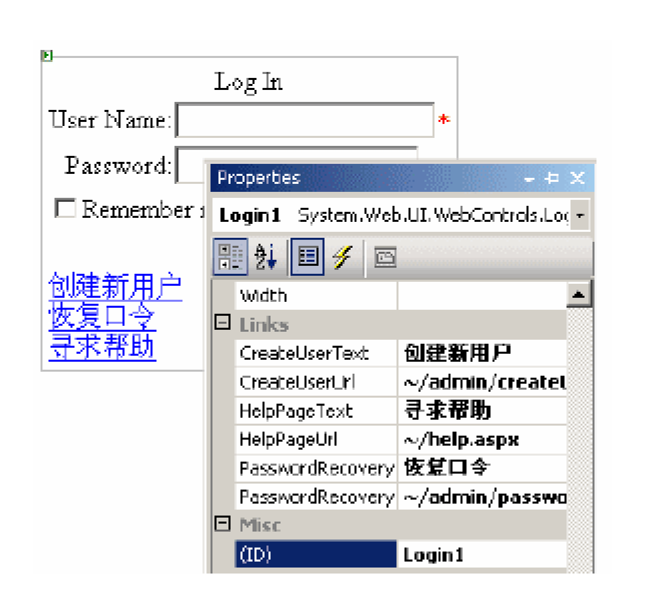


图 21.10 为登录控件设置属性

属性 VisibleWhenLoggedIn 用来设置当客户身份验证成功后是否自动隐藏自己。如果将它设为 true 时，一旦登录成功，Login 控件自己将被隐藏起来。

在 Login 控件中提供了 4 个事件，利用这些事件可以增强控件的功能。

- BeforeLogin：此事件发生在登录表验证之前。利用这一事件可以检查输入数据的语法和格式是否正确，以便及时提示错误并中断后续的操作。
- AfterLogin：该事件发生在认证成功之后，这使你能够在登录成功以后附加一些程序以便做进一步处理。
- Authenticate：该事件发生在当你想根据事件而提供一个固定的认证模式的时候，可以详细说明客户数据是否已经被验证成功。通常，可以利用一个客户个人服务来执行自己的认证机制。

- LoginError：该事件发生在客户输入数据错误，认证停止的时候。利用此事件可以在错误发生、停止认证后做进一步的处理。

21.4.2　使用创建新用户控件

1. 控件的使用方法

利用创建新用户(CreateUserWizard)控件可以在登录表中增添新客户，并为新客户登记相应的参数。将该控件从工具箱拖入网页时将自动生成如图 21.11 所示的界面。

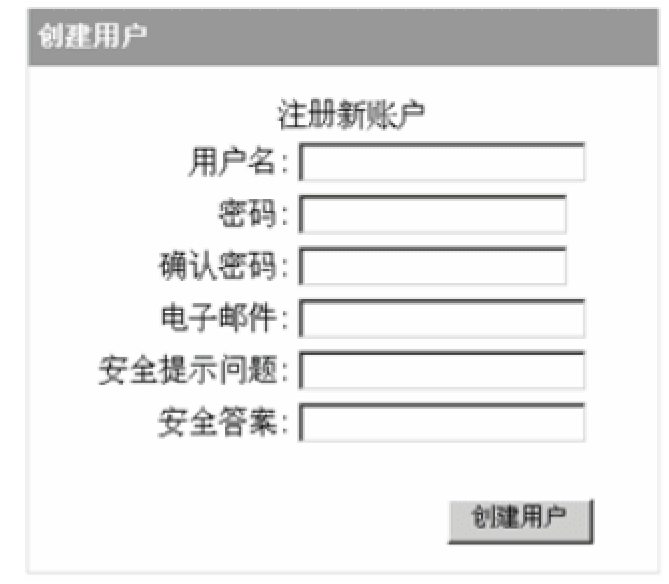


图 21.11　创建新用户界面

控件相应的代码如下。

```
<asp:CreateUserWizard
ID="CreateUserWizard1" Runat="server" />
```

界面中的用户名(User Name)、密码(Password)是识别客户的主要标志。安全提示问题(Security Question)以及安全答案(Security Answer)是为了防止客户忘记自己密码时的提示。

在注册表中，每个客户名都是唯一的，不能与别人重复。对密码的设置有比较严格的要求。为保证密码不容易被人猜中，默认情况下密码的设置要符合“强密码”(Strong Password)的要求。强密码必须是：

- 至少 7 个字符。
- 字符中至少包括一个大写或小写的字母。
- 字符中至少包括一个非数字亦非字母的特殊符号，如“!”、“@”、“#”、“$”、“.”、“,”等。

另外，该控件还有一个强大的功能，就是可以在客户完成所有的注册项目之后，自动给客户的邮箱发送客户注册信息的邮件。比如，可以感谢客户登录你的网站等。这些邮件可以包括客户注册的信息(用户名、密码等)。

可以通过给该控件的 MailDefinition 属性赋值来配置邮件的发送，这个属性代表了 MailDefinition 类的一个对象，MailDefinition 类包括了定义一封 E-mail 需要的所有的属性，还可以在根目录下建立一个.txt 文件，把文件的路径赋给 MailDefinition 属性的 BodyFileName，比如文件名为 welcomeEmail.txt。该文件中还可以包含一些特殊的符号，如<%username%>和<%userpassword%>，用来代替实际的用户名和密码。例如：

```
欢迎您登录本网站
您的名字是： <%username%>
您的密码是： <%userpassword%>
```

下面是在 CreateUserWizard 控件中，为客户完成注册后发送给客户的一封主题为“感谢”的电子邮件而做的设置，邮件的文件名为 welcomeEmail.txt。

```
<asp:CreateUserWizard ID="CreateUserWizard1" Runat="server">
   <MailDefinition >
     BodyFileName="welcomeEmail.txt"
```

```
      From="mySite@tom.com"
      Subject="感谢!"
    </MailDefinition>
</asp:CreateUserWizard>
```

如果要使该控件具有发送电子邮件的功能，必须使它能够发送邮件，为此在 machine.config 文件中将看到下面的设置。

```
<smtpMail
  serverName="localhost"
  serverPort="25"
/>
```

如果安装并激活了一个本地 SMTP 服务，那么就不需要修改上面的设置，如果想用不同的邮件协议，则必须在 Web.config 文件中修改上面的设置。

CreateUserWizard 控件在一些复杂的注册场合中也很有用，比如在授权客户登录你的网站前要验证客户注册的电子邮箱地址是否有效。如果激活了 CreateUserWizard 控件的 AutoGeneratePassword 属性，那么控件就会为客户随机地生成一个密码。利用 CreateUserWizard 控件的 E-mail 功能，将这个随机生成的密码送回给客户，要求客户必须使用这个密码才能登录网站。这样，也就验证了客户邮箱地址的有效性。

上述界面中的有些部分实际上取决于在 Membership Provider 中的设置。例如 Question 和 Answer 文本框只有当 Membership Provider 中的 requiresQuestionAndAnswer 属性设置为 true 时(默认时即如此)才会显示出来。

当这些设置完成后打开 Access 数据库中的客户表，可以看到刚刚建立的客户信息已经存储在数据表中了，在这里并没有编写一行代码。这些复杂的功能实际上都被封装到了 CreateUser 控件之中。

2. 对密码要求的设置

对强密码的上述规定是系统在 machine.config 文件中设置的。默认情况下，所有利用 CreateUserWizard 控件注册的客户都必须按照这个设置执行。如果设计人员想要降低或提高对密码的要求时，可以在自己网站根目录下的 Web.config 文件中重新进行设置。新的设置将在本网站范围内自动覆盖 machine.config 中的设置。

在 machine.config 文件中相关的设置如下。

```
<membership>
   <providers>
      <add name="AspNetSqlMembershipProvider"
          type="System.Web.Security.SqlMembershipProvider
          enablePasswordRetrieval="false"
          enablePasswordReset="true"
          requiresQuestionAndAnswer="true"
          applicationName="/"
          requiresUniqueEmail="false"
          passwordFormat="Hashed"
          maxInvalidPasswordAttempts="5"
          minRequiredPasswordLength="7"
          minRequiredNonalphanumericCharacters="1"
          passwordAttemptWindow="10"
```

```
          passwordStrengthRegularExpression="" />
   </providers>
 </membership>
```

设置中各属性的含义如下。

- enablePasswordRetrieval：是否允许客户检索自己的密码。
- enablePasswordReset：是否允许客户重置自己的密码。当设置为 true 时，一旦客户遗忘了代码，只能用随机产生的代码来替换原来的代码。
- requiresQuestionAndAnswer：是否需要设置向客户提问并要求客户回答的项目。
- passwordFormat：密码加密的格式。
- minRequiredPasswordLength：最小的密码长度。
- minRequiredNonalphanumericCharacters：最少包括非英文字母和数字的特别字符数。
- maxInvalidPasswordAttempts：允许反复实验密码的最多次数。
- passwordAttemptWindow：允许反复实验密码的时间(分钟)。
- passwordStrengthRegularExpression：密码计算的正则表达式。

为了防止恶意顾客用反复实验的方法套取密码，系统设置了 maxInvalidPasswordAttempts 和 passwordAttemptWindow 两个属性。前一项属性规定了最多允许实验次数，后一项规定了允许反复实验的时间(分钟)。如果客户在允许时间内实验次数没有达到允许的次数时，跟踪系统将自动将访问次数归 0；若达到或超过了允许次数时，将被访问客户的 IsLockOut 属性设为 true，以禁止网络对它的访问，直到由管理员调用 UnlockUser 方法解除对该客户的锁定为止。

解锁只能由服务器管理人员进行。在解锁的网页中只需放入一个文本框(TextBox)用来输入客户名。另外再放入一个按钮，在按钮的 Click 事件中编写以下代码。

```
protected void Button1_Click(object sender, EventArgs e)
    {
       MembershipUser Unlock = Membership.GetUser(TextBox1.Text);
       if (Unlock == null)
          Response.Write("没有这个客户。");
       else
       {
          Unlock.UnlockUser();
          Response.Write("该客户已经解锁");
       }
    }
```

下面假定创建一个用于学习的网站，为了简化操作，想降低对密码的要求，为此在网站根目录下的 Web.config 文件中，对<membership>的属性做了以下设置。

```
<membership defaultProvider="AspNetSqlMembershipProvider"
userIsOnlineTimeWindow="15" hashAlgorithmType="" >
        <providers>
          <clear/>
          <add  connectionStringName="LocalSqlServer"
                requiresQuestionAndAnswer="false"
                minRequiredPasswordLength="1"
                minRequiredNonalphanumericCharacters="0"
```

```
                name="AspNetSqlMembershipProvider"
                type="System.Web.Security.SqlMembershipProvider,
System.Web, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" />
        </providers>
      </membership>
```

有了这个设置，当在网站中创建新客户时，CreateUserWizard 控件中将不包括提问和要求回答项(因为 requiresQuestionAndAnswer 属性设为 false)。密码只需一个以上字符，密码中不再需要非英文字母或数字的特别字符。

21.4.3 登录状态与登录姓名控件

一般的登录模块，当客户成功登录后，会显示客户当前登录的身份，比如“欢迎×××客户登录”的提示，同时会显示“Logout(退出)”的提示。这里可以利用 LoginName 和 LoginStatus 控件的帮助来实现这一功能。

LoginName 用来显示注册客户的名字，通过 FormatString 属性可以增加一些格式的描述。如果客户没有被认证，这个控件就不会在页面上产生任何输出。而 LoginStatus(登录状态)控件则提供了一个方便的超链接，它会根据当前验证的状态，在登录和退出操作之间进行切换，如果客户尚未经过身份验证，则显示指向登录页面的链接。如果客户已经进行了身份验证，则显示使该客户能够退出的链接。利用不同的属性，这两个显示的内容都是可以被修改的。通常可以根据登录和退出的状态在控件上加上照片等个性化的东西。

这两个控件产生的对应代码分别如下。

```
<asp:LoginName ID="LoginName1" Runat="server" />
<asp:LoginStatus ID="LoginStatus1" Runat="server" />
```

在 LoginStatus 控件中为了能够正确退出，还需要对下面两个属性进行设置。

- LogoutAction 属性：设成 Redirect(默认是 Refresh)。
- LogoutPageUrl 属性：指定退出的网页，通常是用于登录的网页。

21.4.4 登录视图控件

在早期的版本里，区分不同角色、浏览不同页面需要用代码来实现，这样做比较麻烦。现在 ASP.NET 3.5 提供了一个十分有用的控件，就是 LoginView。LoginView 结合导航控件能够根据当前客户的角色自动显示不同的导航界面，实现基于角色的网站浏览功能。默认情况下该控件只包括两个模板：匿名(未登录)模板(Anonymous)与已登录模板(LoggedIn)，可以对匿名客户和已登录的客户分别显示不同的导航界面。如果在应用项目中设置了多个不同的角色时，控件将自动增加多种不同的模板，用来为不同角色显示不同的导航界面。每个登录后的客户将只能按照自己所充当的角色查看自己权限以内可以访问的网页，从而可以直观地保护网页。然而这只是视图上的保护，并不能代替 Web.config 文件的作用，一些客户还有可能直接利用 URL 直接打开受保护的网页。因此视图的保护还应该和 Web.config 相结合才能既有效又方便地保护网页。

下面用一个简单的示例来说明 LoginView 控件的使用方法。

(1) 将 LoginView 控件拖入窗体，单击【编辑 RoleGroups】按钮，打开角色组编辑对话框，并将已经设置的角色增加到对话框中，如图 21.12 所示。

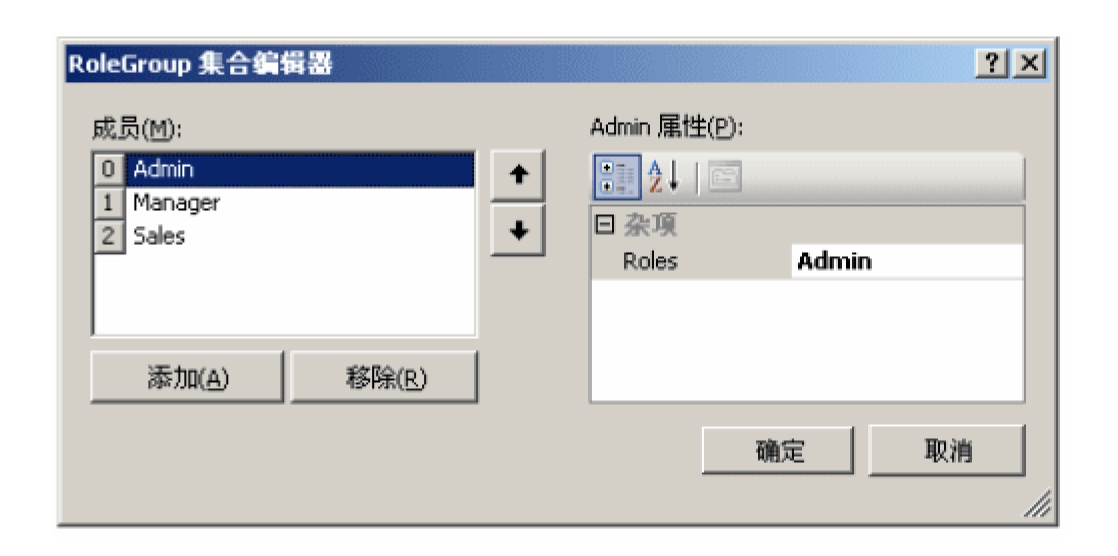


图 21.12　角色组编辑对话框

(2) 查看 LoginView 控件的模板时，将看见除原来的两个模板以外又增加了几个角色的模板，如图 21.13 所示。

选择不同的模板，放入 TreeView 控件，分别按照角色的权限显示相应的网页。

(1) 匿名客户的模板(AnonymousTemplate)如图 21.14 所示。

AnonymousTemplate
LoggedInTemplate
RoleGroup[0] - Admin
RoleGroup[1] - Manager
RoleGroup[2] - sales

图 21.13　多角色的模板

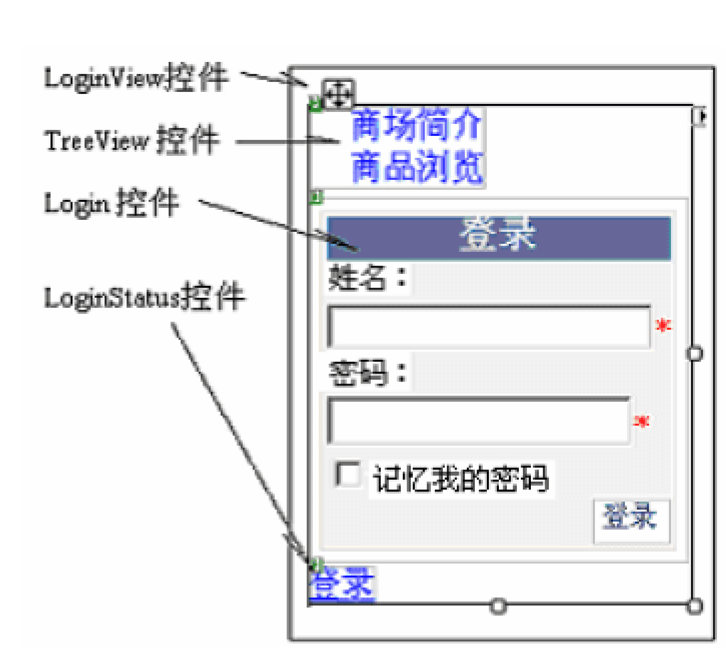


图 21.14　匿名客户的模板

(2) 角色为 Admin 的模板(RoleGroup[0]-Admin)如图 21.15 所示。

(3) 角色为 Sales 的模板(RoleGroup[2]-Sales)如图 21.16 所示。

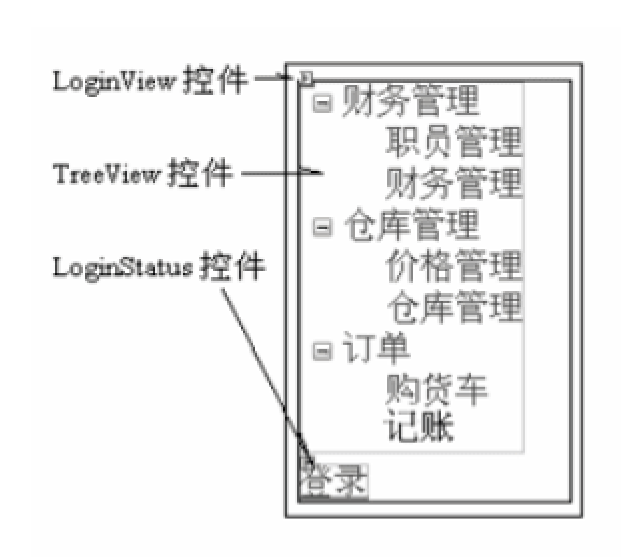


图 21.15　角色为 Admin 的模板

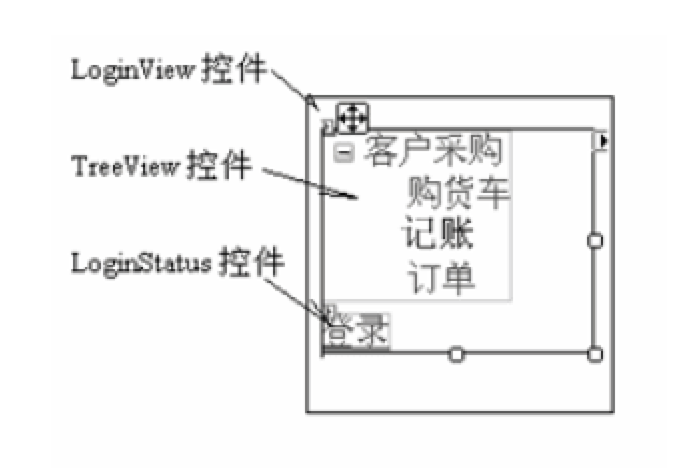


图 21.16　角色为 Sales 的模板

注意：LoginStatus 控件要放在 LoginView 范围之外。

21.4.5 PasswordRecovery 控件和 ChangePassword 控件

有开发经验的人都知道，以前如果客户忘记了密码而要重新获得密码是比较麻烦的事情，为了实现这一功能，需要编写代码，包括认证客户、查询数据库、修改数据库等。ASP.NET 3.5 提供了一个很有吸引力的控件，即 PasswordRecovery 控件。该控件能够通过电子邮件来帮助恢复忘记的密码。

要使用该控件，需要具有自动发送邮件的功能，必须像 CreateUserWizard 控件一样，正确配置 Web.config 文件。

只要客户在注册时正确地填写了邮箱地址和配置正确，并在该控件里提交了请求，它就会自动把密码发送到你的邮箱中。就像 CreateUserWizard 控件一样，也可以用 MailDefinition 属性来定义发送给客户的电子邮件的属性。

此控件提供了 3 种模板。

- UserName：用于初始化控件，客户需要在这里填上登录名。
- Question：在客户寻找遗忘的密码时必须回答的问题。
- Answer：用在当客户输入的密码正确，或者已经用 E-mail 发给客户的时候。

在 PasswordRecovery 控件中还有一些重要的事件。

- BeforeUserLookup：当客户查找客户资料的时候被激发。可以设定个人测试条件取消这个过程。
- UserLookupError：当客户名不存在时激发。
- BeforeAnswerLookup：在客户输入了答案并且被验证后激发。
- AnswerLookupError：当输入答案错误时被激发。
- BeforeSendMail：在邮件发送之前被激发。

Changepassword 控件的用法和 PasswordRecovery 的相似，它也有 MailDefinition 属性，通过设置该属性可以设置发送给客户的邮件格式。

修改密码(Changepassword)控件的界面如图 21.17 所示。

恢复密码(PasswordRecovery)控件的界面如图 21.18 所示。

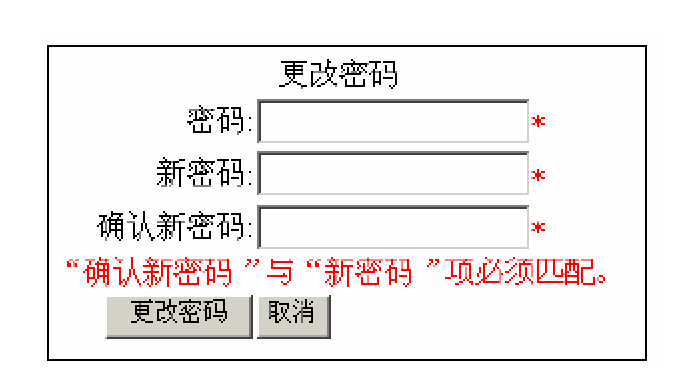


图 21.17 修改密码控件的界面

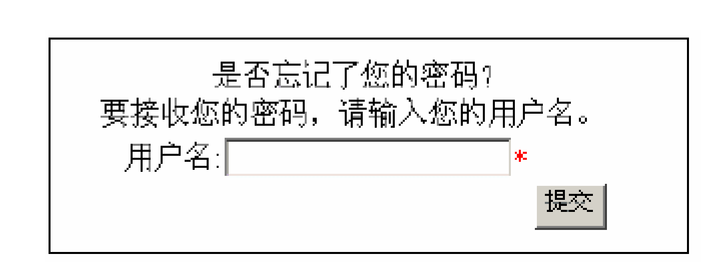


图 21.18 恢复密码控件的界面

21.4.6 在 Login 控件中增添图片校验码

为了在 Login 控件中增添图片校验码，以拒绝机器人行为，需按照以下步骤进行。

(1) 将 Login 控件拖入网页，并将控件转换为模板。

(2) 在模板中增加校验码文本框(TextBox1)以及图片控件。

(3) 网站中增加为图片校验需要的文件并建立它们之间的联系。

(4) 为 Login 控件中按钮的 Click 事件编写代码。

● 当登录成功转向 DestinationPageUrl 属性指定的网页时：

```
protected void Login1_Authenticate(object sender,
AuthenticateEventArgs e)
    {
        string yzcode=(string)Session["CheckCode"];
        TestBox tt = (TextBox)Login1.FindControl("TextBox1");
        if (yzcode != tt.Text)
        {
            Response.Redirect("ErrorMessage.aspx");
        }
}
```

● 当登录成功转向 LoginView1 控件的角色模板时：

```
protected void Login1_Authenticate(object sender, AuthenticateEventArgs e)
    {
        string yzcode=(string)Session["CheckCode"];
        Login LL = (Login)LoginView1.FindControl("Login1");
                //因为 Login1 控件放在 LoginView1 控件中
        TestBox tt = (TextBox)LL.FindControl("TextBox1");
        if (yzcode != tt.Text)
        {
            Response.Redirect("ErrorMessage.aspx");
        }
    }
```

21.5　直接调用 Membership API 方法

为了对成员身份进行更高级别的控制，可以直接使用 Membership API 方法。在 System.Web.Security 命名空间中主要包括两个类：Roles 类和 RolePrincipal 类。

Membership API 是 Membership 类中公有的方法，利用这些方法能够完成以下工作：创建新客户；更改密码；搜索与特定条件匹配的客户；创建角色(CreateRole)；删除角色(DeleteRole)；读取所有角色(GetAllRoles)；读取某个客户分配的角色(GetUsersInRole)和读取某个角色的客户(GetRolesForUser)等。实际上，前面所说的客户管理控件就是使用这些方法来实现客户管理的。

下面举例说明 Membership 类的一些比较常用的方法。

21.5.1　创建新客户

打开网页窗体，在其中添加 6 个文本框以及一个按钮，其中：

- TextBox1 用于输入新客户名。
- TextBox2 用于输入新客户密码。
- TextBox3 用于输入新客户的 E-mail。
- TextBox4 用于输入安全提示问题。
- TextBox5 用于输入回答问题。

- TextBox6 用于输入提示错误。

按钮的代码如下。

```
void Button1_Click(object sender, EventArgs e)
  {
   if (this.IsValid)
     {
      MembershipCreateStatus status;
      MembershipUser user=
Membership.CreateUser(this.TextBox1.Text,this.TextBox2.Text,this.
TextBox3.Text,this.TextBox4.Text, this.TextBox5.Text, false,out status);
      switch (status)
     {
   case MembershipCreateStatus.Success:
       FormsAuthentication.RedirectFromLoginPage(user.UserName, true);
       // 上面代码的最后用 true 时，代表创建成功后自动返回到 default.aspx 网页
       break;
   case MembershipCreateStatus.DuplicateEmail:
       this.TextBox6.Text = "E-mail 地址已经登录";
       break;
   case MembershipCreateStatus.DuplicateUserName:
       this.TextBox6.Text = "客户已经登录";
       break;
       case MembershipCreateStatus.InvalidEmail:
       this.TextBox6.Text = " E-mail 地址格式不对";
       break;
       case MembershipCreateStatus.InvalidPassword:
       this.TextBox6.Text = "密码不对";
       break;
       case MembershipCreateStatus.UserRejected:
       this.TextBox6.Text = "登录失败,原因不清楚" ;
       break;
      }
   }
}
```

注意：在代码的命名空间中要增加“using System.Web.Security;”的引用。

21.5.2 创建新角色

创建新角色的代码如下。

```
void Button1_Click(object sender, EventArgs e)
{
    if (this.TextBox1.Text.Length > 0)
      {
        Roles.CreateRole(this.TextBox1.Text);
      }
}
```

可以通过 Global.asax 文件的 Application_Start()函数，在应用程序首次运行时就自动创建新的角色。为此需要编写如下代码。

```
void Application_Start(object sender, EventArgs e) {
        if (!Roles.RoleExists("Administrators"))
Roles.CreateRole("Administrators");
        if (!Roles.RoleExists("Friends")) Roles.CreateRole("Friends");
    }
```

在这段代码中，先利用 Roles.RoleExists()方法来判断，某个角色是否存在。只有当该角色不存在时，才创建它。在这段代码中创建了 Administrators 和 Friends 两个角色。

注意：只有在【网站管理工具】的【安全】选项卡中选中【角色】后，才能运行这段程序。

21.5.3　给客户分配角色

下面给客户分配角色时不使用 SqlDataSource 数据源控件，数据绑定通过代码实现。给客户分配角色的关键语句是

```
Roles.AddUserToRole(this.DropDownList1.SelectedItem.Text,
                    this.DropDownList2.SelectedItem.Text);
```

其中 DropDownList1 和 DropDownList2 是两个下拉菜单，分别与数据表中的 userName 和角色列表中的 roleName 进行数据绑定。现在举例说明如下。

(1)　在网页中增添相应的控件，其界面如图 21.19 所示。

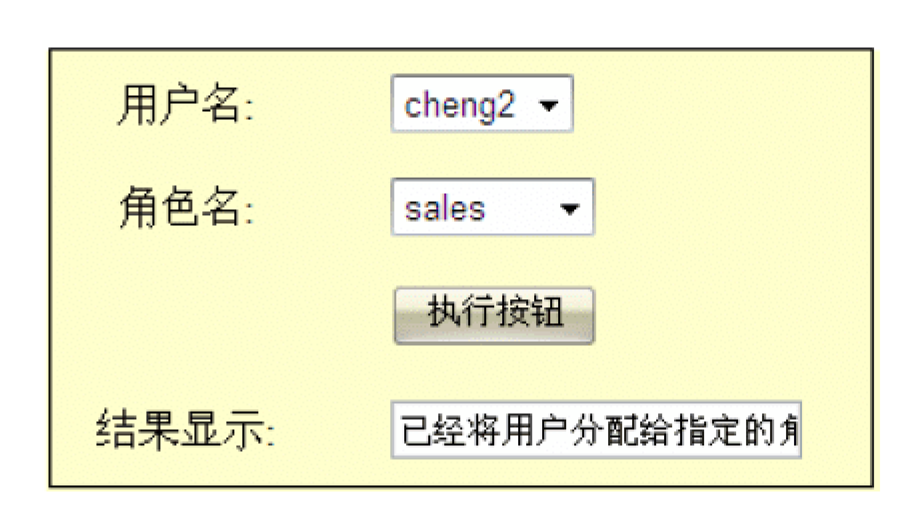


图 21.19　给客户分配角色

(2)　给两个下拉菜单确定数据源。

```
protected void Page_Load(object sender, EventArgs e)
   {
       TextBox1.Text = "";
       string[] rolesArray;
       MembershipUserCollection users;

       if (!IsPostBack)
       {
          //将客户的列表作为 DropDown1 控件的数据源
          users = Membership.GetAllUsers();
          DropDownList1.DataSource = users;
          DropDownList1.DataBind();

          //将角色的列表作为 DropDown2 控件的数据源
```

```
                rolesArray = Roles.GetAllRoles();
                DropDownList2.DataSource = rolesArray;
                DropDownList2.DataBind();
            }
    }
```

(3) 调用 Membership API 方法给客户分配角色。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        // 指定客户及被分配的角色

        if (DropDownList1.SelectedItem == null)
        {
            TextBox1.Text = "Please select a user.";
            return;
        }
        if (DropDownList2.SelectedItem == null)
        {
            TextBox1.Text = "Please select a role.";
            return;
        }

        //将客户分配给指定的角色
        try
        {
            Roles.AddUserToRole(DropDownList1.SelectedItem.ToString(),
DropDownList2.SelectedItem.ToString());
            TextBox1.Text = "已经将客户分配给指定的角色.";
        }
        catch
        {
            // 若重复分配时，将提示错误
            TextBox1.Text = "重复分配错误";
        }
    }
```

21.5.4 删除角色

为了删除角色，需要在网页中设置一个下拉菜单与一个按钮，并将角色的列表作为数据源与下拉菜单进行数据绑定。按钮的事件代码如下。

```
void Button1_Click(object sender, EventArgs e)
   {
      Roles.DeleteRole(this.DropDownList2.SelectedItem.Text);
   }
```

注意：只有没有分配给客户的角色才能够被删除。否则需要先删除相关的客户，然后才能删除给这些客户分配的角色。

21.5.5 从角色中删除客户

为了删除客户，需要在网页中设置两个下拉菜单与一个按钮，分别将客户的列表和角

色的列表作为数据源与各自的下拉菜单进行数据绑定。按钮的事件代码如下。

```
void Button1_Click(object sender, EventArgs e)
  {
   Roles.RemoveUserFromRole(this.DropDownList1.SelectedItem.ToString(),
                            this.DropDownList2.SelectedItem.ToString());
  }
```

21.5.6　删除客户

在注册表中删除客户的代码如下。

```
void Button1_Click(object sender, EventArgs e)
   {
    Membership.DeleteUser(string userName, bool deleteAllRelatedData);
   }
```

21.6　小　　结

基于角色的安全技术目前已经成为网站的普遍需要。设计基于角色的安全技术程序，特别是设计一套功能完备的保护程序是一项艰巨的任务。但是由于这些程序的功能和模式都相对稳定，因此系统有可能为设计者完成大部分功能。.NET 框架就是根据这一特点，对传统方法做了进一步的封装和抽象，提供了智能工具和组合控件，将客户管理和网页安全方面的大部分复杂工作，包括数据表的生成、连接、添入数据和查询等都隐藏在内部自动进行，这就大大简化了设计过程。

在系统强力的支持下，现在实行基于角色的安全技术，主要是将网页进行合理的分类，并放置于不同的目录下，然后用 Web.config 保护目录下的文件。可以直接在 Web.config 文件中撰写保护策略，也可以利用网站管理工具或者调用 Membership API 方法写入。

系统提供的 7 个控件一旦生成，就具备基本的显示界面和比较完善的功能，设计者只需根据情况进行设置和修改，以符合应用程序的实际需要。

21.7　习　　题

1. 填空题

(1)　LoginStatus 控件用来显示客户的__________，以便可以随时退出登录状态。

(2)　LoginName 控件用来自动显示登录的__________。

(3)　当利用 CreateUserWizard 控件创建新客户时，密码不能随便设置，必须符合以下 3 项条件：__________；__________；__________。

(4)　帮助客户恢复密码可以利用__________控件进行设计。

(5)　帮助客户修改密码可以利用__________控件进行设计。

2. 选择题

(1) 为了保护网页，需要先将被保护的网页分类放在不同的子目录下，这是为了________。

A. 便于管理　　B. 调用方便
C. 集中网页的入口　　D. 便于网站迁移

(2) 在一个子目录的 Web.config 文件中有如下一段代码。

```
<authorization>
    <allow roles="admin" />
    <allow roles="manager" />
    <deny users="*" />
    <allow roles="sales" />
</authorization>
```

允许访问此子目录下的网页的角色有__________。

A. admin　　B. manager
C. admin 和 manager　　D. admin、manager 和 sales

(3) 客户登录控件(Login)中的 DestinationPageUrl 属性代表___________。

A. 登录成功的提示　　B. 登录成功时转向的网页
C. 登录失败时转向的网页　　D. 登录失败时的提示

3. 判断题

(1) 所谓角色是若干具有相同访问权限客户的集合。 (　)
(2) 只能给每个客户分配一个角色。 (　)
(3) 登录视图控件(LoginView)只能有两种模板，因而只能载入两种视图。 (　)

4. 简答题

(1) 简述 ASP.NET 3.5 对网页保护设计的支持。
(2) 简述利用 ASP.NET 网站管理工具定义角色、创建客户和指定访问规则的步骤。

5. 操作题

(1) 在系统中设置三种角色并分配给多个客户，他们的权限各不相同；在登录视图控件(LoginView)中为这三种角色设置不同的网站视图，以实现不同角色访问网站时的不同结果。

(2) 在网页中设置控件，编写代码利用 Memership API 来创建新客户。
(3) 在网页中设置控件，编写代码利用 Memership API 来创建新角色。
(4) 在网页中设置控件，编写代码利用 Memership API 来删除角色。
(5) 在网页中设置控件，编写代码利用 Memership API 给客户分配角色。
(6) 在网页中设置控件，编写代码利用 Memership API 从角色中删除客户。

第 22 章　网站的个性化服务

网站的个性化服务是近几年才发展起来的一项热门技术，也是当前网站竞争的焦点之一。ASP.NET 3.5 为个性化服务设计提供了强有力的支持。本章将讲述如何在系统的支持下进行个性化设计的方法，主要包括三方面的问题：个性化服务的基础知识、保留客户关心的数据以及创建个性化主页的方法。具体问题包括：

- 概述。
- ASP.NET 3.5 对个性化设计的支持。
- 保留客户关心的数据。
- Web art 介绍。
- 创建个性化主页。

22.1　概　　述

进入 21 世纪以来，一些大型网站如 Yahoo！、Google、Live 等纷纷推出了“个性化主页”等个性化服务措施，吸引着广大的网民。人们惊呼一个网站个性化服务的时代已经来临。

网站个性化服务技术迅猛发展的历史背景是：随着网站功能的扩大和增强，客户的需求也变得更加多样化起来。例如，有的客户想参与股票交易，有的喜爱体育新闻，有的爱读时事评论等。就以参加远程教育的网民来说，第一次进入网站与后续进入网站的目的也不相同。第一次进入时，常常是为了全面地了解信息，后续进入时，则常常是为了了解当天的教学进度等。

客户们带着不同的目的进入网站，但是却有一个共同点，就是都希望能够立即进入自己喜爱的环境，迅速地找到自己关心的信息。作为网站的开发者如何来满足客户的这个共同的愿望呢？唯一的方法就是实现网站的个性化服务。

网站的个性化服务实质上就是给客户提供自己定制环境的能力和机会，用通俗的话来比喻就是给客户开设“自助餐”。允许客户自己部署网页界面，创建自己喜爱的环境，保留自己关心的数据。系统则保存这些设置并且将这些设置与客户联系起来，一旦客户再次打开网站，就将这些设置重现在客户面前。

为了实现这几点，系统必须具备以下能力。

- 识别和保存客户信息。
- 允许客户自己定制网页界面。
- 自动保存客户定制的界面和关心的数据。
- 跟踪客户，以便当客户再次出现时将这些设置反馈给客户。

若用传统设计方法来满足上述要求将非常困难，现在 ASP.NET 3.5 为个性化设计提供了强大的支持，从而大大简化了设计的过程。

22.2　ASP.NET 3.5 对个性化设计的支持

在 ASP.NET 3.5 中，系统对个性化服务设计提供的支持包括三大部分：Membership、profile 和 WebPart。

- Membership：用于识别和管理客户信息。
- profile：用于确定和保存客户关心的数据。
- Web Part：用于给客户定制自己的网页界面。

22.2.1　关于 Membership

Membership 用来识别和管理客户。客户管理与个性化服务密不可分，如果不能识别客户，个性化服务就无从谈起。因此可以说客户管理是个性化服务的基础。

关于如何识别客户在第 18 章中已经有过比较详细的叙述。按照 ASP.NET 3.5 系统的要求，首先要利用网站管理工具进行配置，以便在 App_Data 目录下自动生成专用的数据库和数据表以存储客户的信息。另外还需要利用客户登录、分配角色、分配权限等一系列操作来完成对客户的分类和授权工作(具体做法参考第 18 章)。这些工作实际上就为识别客户、跟踪客户奠定了基础。

除此以外还应看到，进入网站的客户中还有相当一大批是匿名客户，这些客户没有经过登录就进入网站，对这些客户应不应该进行个性化服务呢？如果不考虑这些客户，可能就会因此而失去一大批网民。因此网站的个性化服务不能只面向一部分客户，而应该面向客户的全体。

既然这样，如何来识别匿名客户呢？在这里，系统采用了一种非常巧妙的方法来识别他们。这就是每当匿名客户来访时，就自动给该匿名客户设置一个标志，并且利用 Session 与 Cookie 对象相结合来识别他们。如果客户封掉了浏览器 Cookie 的使用时，也可以利用不使用 Cookie 情况下识别客户的方法(具体做法参考第 8 章)。在这里，设计者只需要在 Web.config 文件中进行一些简单的设置，系统就会自动完成对匿名客户的识别工作。设置的语句如下。

```
<system.web>
    <anonymousIdentification  enabled="true"/>
      …
</system.web>
```

22.2.2　关于 profile

1. 什么是 profile

profile 是一组为特定客户定义且可以保存的数据。一个 profile 可以是一个简单的信息(例如客户名)，也可以是一个比较复杂的数据对象。profile 以 XML 代码的形式定义并存储在 Web.config 文件中。每个 profile 要尽量用最简略的形式来定义，通常只包括名和类型两项。如果没有类型定义时，使用默认类型 string。

2. 简单的 profile 的定义方法

定义 profile 的方法举例说明如下。

```
<profile>
  <property name="NumberOfApples" type="int"/>
</profile>
```

这里定义了一个名为 NumberOfApples 的 profile，它的类型是整型数(int)。如果这个 profile 也允许匿名客户使用时，还应该在 property 的附加属性中加以说明。例如：

```
<system.web>
    <anonymousIdentification  enabled="true"/>
    <profile>
      <properties>
        <add name=" NumberOfApples " type=" int "
                   allowAnonymous="true" />
      </properties>
    </profile>
</system.web>
```

3. 拥有成组属性的 profile 的定义方法

如果一个 profile 包括有较多的属性时，可以分组进行定义。定义的方法举例如下。

```
<profile>
 <properties>
   <group name="Preferences">
     <add name="ShowQuoteOfTheDay"
        defaultValue="true" type="System.Boolean" />
     <add name="ShowNews" type="System.Boolean" />
</group>
<group name="BillingAddress">
     <add name="Street" type="System.String" />
     <add name="City"  defaultValue="Toronto" type="System.String" />
<add name="StateProv" type="System.String" />
<add name="ZipPostal" type="System.String" />
</group>
</properties>
</profile>
```

代码中的 profile 包括两组属性，用两个 group 来定义。注意，各个组名不能重复，每个 group 内部的名字也不能重复，但是各个 group 之间内部的名字允许重复。

4. 读取 profile

只要在 Web.config 文件中正确地定义了 profile 属性，并且经过编译，该属性就会被组合到系统中，因而可以立即被应用程序所调用，可以像读取某个控件的属性一样来读取 profile 属性的值。

例如使用前面的分组定义，当调用 Profile.BillingAddress(组名)以后增加一个句号时，将弹出该对象的所有属性，如图 22.1 所示。

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Label1.Text=Profile.BillingAddress.|
    }
    protected void Button1_Click(object
    {

    }
}
```

```
City
Equals
GetHashCode
GetPropertyValue
GetType
Init
SetPropertyValue
StateProv
Street
ToString
```

图 22.1 自动弹出 profile 中设置的属性

5. 关于 profile provider

ASP.NET 3.5 对 profile 的管理是自动进行的，系统提供的 profile provider 能将客户输入的 profile 数据通过 web.config 文件的设置，自动保存到专用数据库(ASPNETDB.MDF)的专用数据表(aspnet_Profile)中，还能将这些数据与客户的标志紧密地联系在一起。其过程如图 22.2 所示。

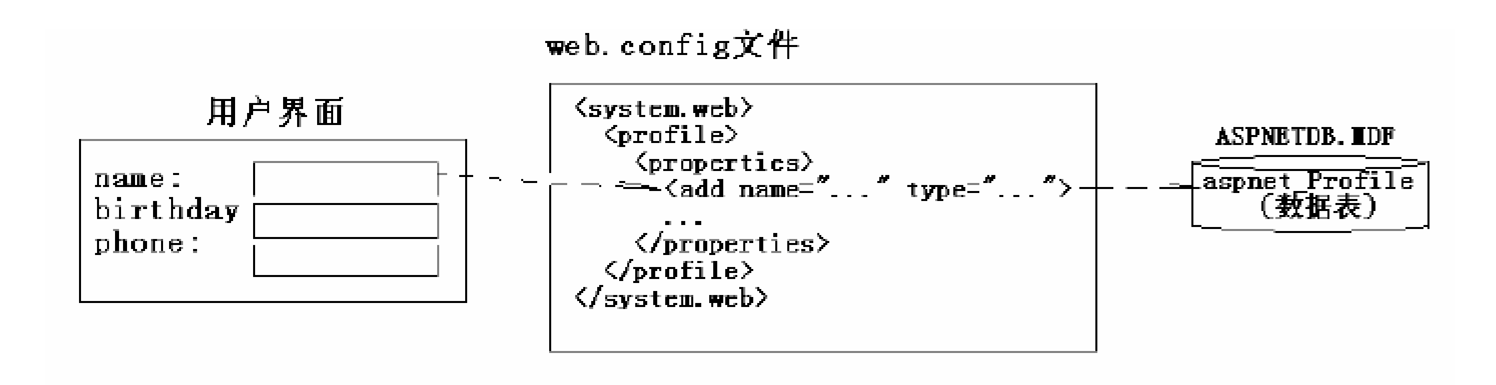


图 22.2 数据库中自动保存 profile 信息

Profile 对象与 Session 对象十分相似，但是更好用一些。与 Session 对象相似的地方在于，Profile 对象是相对于一个特定的客户的，也就是说，每个 Web 应用程序的客户都有他们自己的 Profile 对象。与 Session 对象不同的是，Profile 对象是持久对象。如果你向 Session 对象中添加一个项，在你离开网站时，该项就会消失。而 Profile 对象则完全不同，当你修改 Profile 对象的状态时，修改在多个访问之间均有效。

profile 使用 provider 模式来存储信息，默认情况下，user profile 的内容会保存在 SQL Server Express 数据库中，该数据库位于网站的 App_Data 目录下。

与 Session 不同，Profile 对象是强类型的，Session 对象仅仅是一个项集合而已，而 Profile 对象则有强类型属性。

使用强类型有它的好处。例如，使用强类型，就可以在 Microsoft Visual Web Developer 中使用智能感知技术，当输入 profile 和一个点的时候，智能感知会弹出你已经定义过的 profile 属性列表。

22.2.3　关于 WebPart

Web Part 是系统为客户组建网页而提供的控件。ASP.NET 3.5 提供的 WebPart 分成三类：Web 控件管理器(WebPartManager)、控件带(WebPartZone)和控件(WebPart)。三者之间存在着层次关系。

通常情况下，设计者在网页中放置多种 WebPart，允许客户选择哪些显示，哪些不显示，整个界面如何排列等，以构建自己喜爱的环境。详细情况在后文讲述。

22.3　保留客户关心的数据

下面用两个简单的示例来说明通过个性化服务让客户保留自己关心的数据的方法。

例 22.1　为客户保留电话号码(PhoneCode)。

假定有客户想在网页上保留某个电话号码，每次打开网页时就能够看到这个号码。设计的步骤如下。

(1) 在 Web.config 文件的 profile 中定义 PhoneCode 对象，并允许匿名客户使用这个对象。定义的语句如下。

```
<system.web>
    <anonymousIdentification  enabled="true"/>
      <profile>
        <properties>
            <add name=" PhoneCode "type="string" allowAnonymous="true"/>
        </properties>
        </profile>
</system.web>
```

(2) 在网页上设置控件，让客户自己输入需要保留的数据。

现在网页界面中放置一个名为 TextBox1 的文本框控件、一个名为 Button1 的按钮控件和一个名为 Label1 的标签控件，并在按钮的事件中编写如下代码。

```
protected void Button1_Click1(object sender, EventArgs e)
    {
     Profile.PostalCode = Server.HtmlEncode(TextBox1.Text);
     Label1.Text = Profile. PhoneCode;
    }
```

此段代码的含义是将 TextBox1 中输入的数据保存在 Profile.PhoneCode 对象中，并在 Label1 控件中显示出来。程序运行时单击按钮，结果如图 22.3 所示。

实际上这个 Profile.PhoneCode 的值已经自动保存在 App_Data 目录下的专用数据库的 aspnet_Profile 数据表中。

(3) 为了在重新打开网页时显示该数据，在 Page_Load 事件中编写如下代码。

图 22.3　自动记忆电话号码

```
protected    void    Page_Load(object    sender,
EventArgs e)
```

```
{
Label1.Text = Profile. PhoneCode;
}
```

这段代码的含义是网页打开时将电话号码重新显示在 Label1 控件中。这个例子与通常的示例不同之处在于：同一个网页不同的客户打开时，将只显示该客户自己保存的代码，不同的客户打开显示的代码各不相同。这就是个性化服务的奥妙所在。

例 22.2 为客户保留喜爱的网址。

(1) 在 Web.config 文件中定义 Profile 对象，假定将其命名为 FavoriteUrl。语句如下。

```
<system.web>
<anonymousIdentification  enabled="true"/>
<profile>
<properties>
<add name="FavoriteUrls"
type="System.Collections.Specialized.StringCollection"
         allowAnonymous="true" />
</properties>
</profile>
</system.web>
```

比如想记住某个朋友的生日及电话号码时，可以在 Web.config 文件中设置。

```
  <system.web>
      <anonymousIdentification enabled="true"/>
       <profile>
           <properties>
               <group name="Birthday">
                   <add name="Name" type="System.String" allowAnonymous="true"/>
                   <add name="Birthday" type="System.String"
                             allowAnonymous="true"/>
                   <add name="Phone" type="System.String"
                             allowAnonymous="true"/>
               </group>

           </properties>
       </profile>
</system.web>
```

并在某张网页中设置如下代码，将具体数据输入并在数据表中记录下来。

```
protected void Button1_Click(object sender, EventArgs e)
    {
       Profile.Birthday.Name = TextBox1.Text;
       Profile.Birthday.Birthday = TextBox2.Text;
       Profile.Birthday.Phone = TextBox3.Text;
    }
```

而在另一些网页中显示出来。在这些网页中的 Page_Load 事件中写下如下代码。

```
protected void Page_Load(object sender, EventArgs e)
    {
       TextBox1.Text = Profile.Birthday.Name;
```

```
        TextBox2.Text = Profile.Birthday.Birthday;
        TextBox3.Text = Profile.Birthday.Phone;
    }
```

(2) 在网页上设置控件，让客户逐个输入喜爱的网址。

放置的控件包括一个名为 TextBox1 的文本框控件、一个名为 Button1 的按钮控件、一个名为 ListBox1 的列表框控件。TextBox1 控件用于逐个输入网站的 URL，ListBox 控件用于同时显示多个网站的 URL。用于输入 URL 的按钮的 Click 事件的代码如下。

```
protected void Button1_Click1(object sender, EventArgs e)
{
 string urlString = Server.HtmlEncode(TextBox1.Text);
 if (Profile.FavoriteUrls == null)
  {
   Profile.FavoriteUrls = new System.Collections.Specialized.StringCollection();
  }
 Profile.FavoriteUrls.Add(urlString);
 DisplayFavoriteUrls();
}

void DisplayFavoriteUrls()
 {
   ListBox1.DataSource = Profile.FavoriteUrls;
   ListBox1.DataBind();
}
```

(3) 为了恢复数据需要在 Page_Load 事件中编写如下代码。

```
protected void Page_Load(object sender, EventArgs e)
  {
   DisplayFavoriteUrls();
 }
```

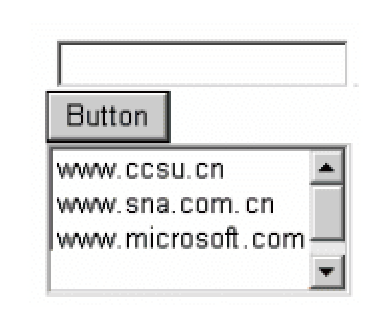


图 22.4　保留网址

程序运行的结果如图 22.4 所示。

为了验证，不妨将定义语句中的 allowAnonymous="true" 部分取消，即系统不再识别匿名客户。其定义变成

```
<properties>
    <add name="FavoriteUrls"
type="System.Collections.Specialized.StringCollection" />
</properties>
```

然后再重复前面的实验，将会看到作为匿名客户重复打开网页时，新增加的网址将不能保持。

22.4　WebPart 介绍

22.4.1　定制网页时能够执行的任务

客户在定制自己的网页时能够执行以下任务：

- 在多个控件中选择显示的控件。
- 按照任意顺序排列控件。
- 改变控件的某些外貌。
- 保存视图直到再次访问。

22.4.2 WebPart 的分类

WebPart 是为客户定制网页界面而提供的控件。在 ASP.NET 3.5 工具箱的 WebPart 选项卡中提供了 13 个控件，这些控件可以分成 3 类。

1. WebPartManager 控件

WebPartManager 在运行中通常不会显示出来，但在幕后它将完成很多工作。它的作用是连接各个网页中的 WebPart，维护和设置各个 WebPart 在网页中的状态。每个定制的网页中必须有一个，也只能有一个 WebPartManager。而且在设计可定制的网页时，第一个放入网页的控件必须是 WebPartManager 控件。

2. WebPartZone 控件

WebPartZone 是 WebPart 的外包装，用来确定 WebPart 在网页中的位置以及它们的外观，正是 WebPartZone 才将 WebPart 分割成一个个独立的可编辑的部分。工具箱提供了 4 种类型的 Zone 控件，它们是 WebPartZone、EditorZone、ConnectionsZone 和 CategoryZone。每种 Zone 中只能放入一定类型的 WebPart。例如 WepPartZone 中可以放入一个或多个服务器控件或者用 GenericWebPart 包围的用户控件；EditorZone 中只能放入能够编辑的控件，如 LayoutEditorPart、AppearanceEditorPart、BehaviorEditorPart 等；CategoryZone 中只能放入 DeclarativeCatalogPart 等。

3. WebPart 控件

可以将 WebPart 看成网页中的模块，将它们放在 WebPartZone 控件中。程序运行时客户可以对其中的 WebPart 进行增添、删除、最小化和恢复等项操作。还可以利用拖放功能改变 WebPart 的网页布局。有的 WebPart 控件中还可以放入通用的服务器控件或者用户控件。

22.5 定 制 主 页

让客户定制自己的主页是当前实现个性化服务的主要内容。系统应该给客户提供这方面的能力和机会。

下面将利用系统在帮助文件中提供的示例，由简到繁地加以介绍，以说明设计的方法。

22.5.1 创建简单的包含 WebPart 控件的网页

1. 将网页划分成几个独立的地带

将网页划分成几个独立地带的步骤如下。

(1) 关闭现有网页，增加一张新网页并将其取名为 WebPartDemo.aspx。

(2) 转向【设计】视图。

(3) 从工具箱的 WebPart 选项卡中将一个 WebPartManager 控件拖入网页中。每个 WebPart 网页必须有一个也只能有一个 WebPartManager 控件，该控件并不提供什么输出。

(4) 在 WebPartManager 控件的下方，利用表格进行布局。这里使用一行三列的布局，将每列的 VAlign(Vertical Align)对齐属性设置为 top。

(5) 将一个 WebPartZone 控件拖入左边的列中，并对其属性做以下设置。

```
ID:SidebarZone
HeaderText:Sidebar
```

(6) 将另一个 WebPartZone 控件拖入到中间的列中，并对其属性做以下设置。

```
ID:MainZone
HeaderText:Main
```

(7) 保存文件。

2. 给各个地带放入内含的控件

现在网页中已经有了两个独立的地带，放置在这些地带中的控件可以单独进行控制。下面就在这些独立的地带中分别放入显示控件。可以放入的控件包括用户控件或者其他服务器控件。当将标准服务器控件放入 WebPartZone 中间时，控件必须放在<zonetemplate>与</zonetemplate>元素之间。只有这样，ASP.NET 在运行时才会把这些控件当做 Web Part 控件对待。

现在首先创建 main zone 中的内含控件。

(1) 打开【设计】视图，将一个 Label 标准控件拖入到 MainZone 控件中。

(2) 打开【源】视图，注意 Label 控件已经在 MainZone 控件的<zonetemplate>与</zonetemplate>元素之间。

(3) 给<asp:label>元素设置属性。首先删除原来的 Text="Label1"属性，增加一些字符串放在<h2>标记之间。其代码如下。

```
<asp:webpartzone id="MainZone" runat="server" headertext="Main">
   <zonetemplate>
     <asp:label id="label1" runat = "server" title="Content">
        <h2>欢迎参观 Web Part 网页</h2>
     </asp:label>
   </zonetemplate>
</asp:webpartzone>
```

(4) 保存文件。

3. 创建用户控件

下面创建一个用户控件作为 WebPart 控件。

(1) 在 WebPartDemo.aspx 网页的同一目录下创建一个用户控件，将其取名为 Searchcontrol.ascx。

(2) 转向【设计】视图，将一个 TextBox 控件与 Button 控件拖入用户控件中。

(3) 转入【源】视图，其代码如下。

```
<%@ control language="C#" classname="SearchControl" %>
<asp:textbox runat="server" id=" textbox1"></textbox1>
<br/>
<asp:button runat="server" id="button1" text="Search" />
```

(4) 保存并关闭文件。

4. 给 Sidebar Zone 增添控件

下面给 Sidebar Zone 内增添两个控件：一个包括一组链接指针；另一个是前面创建的用户控件。

利用 Label 标准控件产生一组指针，除建立超链接以外，其他方法与增加到 main zone 时的方法相同。用户控件则可以直接拖入到 zone 中来，运行时用户控件的外层将自动用 GenericWebPart 控件包装。GenericWebPart 自动成为用户控件的父控件。通过父控件的子控件属性可以调用用户控件中的服务器控件。由于 GenericWebPart 继承于 WebPart 类，因此它与 WebPart 具有相同的属性和方法。

给 Sidebar Zone 增添控件的具体步骤如下。

(1) 打开 WebPartsDemo.aspx 网页。

(2) 转向【设计】视图。

(3) 将前面建立的 SearchControl.ascx 用户控件拖入 SidebarZone 中。

(4) 保存网页。

(5) 转向【源】视图。

(6) 在 SidebarZone 内的<asp:webpartzone>元素中用户控件的上方增加<asp:label>元素标记，包括链接指针。代码如下。

```
<asp:WebPartZone id="SidebarZone" runat="server" headertext="sidebar">
  <zonetemplate>
    <asp:label runat="server" id="linksPart" title=" 我的链接 ">
       <a href=http://www.ccsu.cn> 长沙大学 </a>
       <br/>
<a href=http://www.sina.com.cn> 新浪网站 </a>
       <br/>
     </asp:label>
<uc1:SearchControl     id="searchPart"
runat="server" title="Search" />
</zonetemplate>
</asp:WebPartZone>
```

(7) 保存并关闭文件。

5. 检查网页

在浏览器中查看网页，定制的主页如图 22.5 所示。



图 22.5 定制的简单主页

小方框的右上角出现“▾”下拉标记，单击该

标记，将弹出【最小化】和【关闭】命令。当已经最小化后再单击时，将弹出【还原】命令。

22.5.2　创建可以编辑和改变布局的网页

WebPart 提供了改变网页布局的功能，允许将 WebPart 控件从一个 Zone 拖放到另一个 Zone 中。还允许编辑控件的一些特性，包括它们的外貌、布局和行为。系统还提供了一些基础函数来进行这些工作，并且保持客户自己的编辑结果。

为实现这些功能，下面将在网页中增添一个<asp:editorzone>元素以及两个可编辑的控件。

(1) 增添一个新用户控件，放在 WebPartDemo.aspx 网页的同一个目录下，将用户控件取名为 DisplayMenu.ascx。

(2) 转向用户控件的【源】视图。

(3) 移除现有窗口的代码，将以下代码粘贴到网页中。这是一段 WebPart 控件的用户控件代码，这段代码能够设置和改变显示的模式，并且使用户在该模式中可以改变网页的布局以及显示的外貌。

```
<%@ control language="C#" classname="DisplayModeMenuCS"%>
<script runat="server">

 //Use a field to reference the current WebPartManager control.
  WebPartManager _manager;

  void Page_Init(object sender, EventArgs e)
  {
    Page.InitComplete += new EventHandler(InitComplete);
  }

  void InitComplete(object sender, System.EventArgs e)
  {
    _manager = WebPartManager.GetCurrentWebPartManager(Page);

    String browseModeName = WebPartManager.BrowseDisplayMode.Name;

    //Fill the drop-down list with the names of supported display modes
    foreach (WebPartDisplayMode mode in
      _manager.SupportedDisplayModes)
    {
      String modeName = mode.Name;
      //Make sure a mode is enabled before adding it.
      if (mode.IsEnabled(_manager))
      {
        ListItem item = new ListItem(modeName, modeName);
        DisplayModeDropdown.Items.Add(item);
      }
    }
```

```
    //If Shared scope is allowed for this user, display the
    //scope-switching UI and select the appropriate radio
    //button for the current user scope
    if (_manager.Personalization.CanEnterSharedScope)
    {
      Panel2.Visible = true;
      if (_manager.Personalization.Scope ==
        PersonalizationScope.User)
        RadioButton1.Checked = true;
      else
        RadioButton2.Checked = true;
    }
  }

  //Change the page to the selected display mode
  void DisplayModeDropdown_SelectedIndexChanged(object sender,
    EventArgs e)
  {
    String selectedMode = DisplayModeDropdown.SelectedValue;

    WebPartDisplayMode mode =
     _manager.SupportedDisplayModes[selectedMode];
    if (mode != null)
      _manager.DisplayMode = mode;
  }

  // Set the selected item equal to the current display mode.
  void Page_PreRender(object sender, EventArgs e)
  {
    ListItemCollection items = DisplayModeDropdown.Items;
    int selectedIndex =
      items.IndexOf(items.FindByText(_manager.DisplayMode.Name));
    DisplayModeDropdown.SelectedIndex = selectedIndex;
  }

  //Reset all of a user's personalization data for the page
  protected void LinkButton1_Click(object sender, EventArgs e)
  {
    _manager.Personalization.ResetPersonalizationState();
  }

  //If not in User personalization scope, toggle into it
  protected void RadioButton1_CheckedChanged(object sender, EventArgs e)
  {
    if (_manager.Personalization.Scope ==
      PersonalizationScope.Shared)
      _manager.Personalization.ToggleScope();
  }

  //If not in Shared scope, and if user has permission, toggle
  //the scope
```

```
  protected void RadioButton2_CheckedChanged(object sender,
    EventArgs e)
  {
    if (_manager.Personalization.CanEnterSharedScope &&
        _manager.Personalization.Scope ==
          PersonalizationScope.User)
        _manager.Personalization.ToggleScope();
  }
</script>
<div>
  <asp:Panel ID="Panel1" runat="server"
    Borderwidth="1"
    Width="230"
    BackColor="lightgray"
    Font-Names="Verdana, Arial, Sans Serif" >
    <asp:Label ID="Label1" runat="server"
      Text=" Display Mode"
      Font-Bold="true"
      Font-Size="8"
      Width="120" />
    <asp:DropDownList ID="DisplayModeDropdown" runat="server"
      AutoPostBack="true"
      EnableViewState="false"
      Width="120"
      OnSelectedIndexChanged="DisplayModeDropdown_SelectedIndexChanged"/>
    <asp:LinkButton ID="LinkButton1" runat="server"
      Text="Reset User State"
      ToolTip="Reset the current user's personalization data for
      the page."
      Font-Size="8"
      OnClick="LinkButton1_Click" />
    <asp:Panel ID="Panel2" runat="server"
      GroupingText="Personalization Scope"
      Font-Bold="true"
      Font-Size="8"
      Visible="false" >
      <asp:RadioButton ID="RadioButton1" runat="server"
        Text="User"
        AutoPostBack="true"
        GroupName="Scope"
        OnCheckedChanged="RadioButton1_CheckedChanged" />
      <asp:RadioButton ID="RadioButton2" runat="server"
        Text="Shared"
        AutoPostBack="true"
        GroupName="Scope"
        OnCheckedChanged="RadioButton2_CheckedChanged" />
    </asp:Panel>
  </asp:Panel>
</div>
```

注意：这段代码是系统提供的，系统只在代码中间进行了简要的说明。

(4)　保存用户控件。

(5)　打开 WebPartDemo.aspx 网页，并转向【设计】视图。

(6)　在 WebPartManager 控件的前面留出一行空间，将刚才创建的用户控件拖入其中。

(7)　将 EditorZone 控件拖入布局表格最右边的一列中。

(8)　在 EditorZone 控件中放入 AppearanceEditorPart 控件和 LayoutEditorPart 控件，此时一个可编辑的主页如图 22.6 所示。

(9)　打开网页的【源】视图，将看到以下代码。

```
<td valign="top">
  <asp:EditorZone ID="EditorZone1" runat="server">
    <ZoneTemplate>
      <asp:AppearanceEditorPart ID="AppearanceEditorPart1"
        runat="server" />
      <asp:LayoutEditorPart ID="LayoutEditorPart1" runat="server" />
    </ZoneTemplate>
  </asp:EditorZone>
</td>
```

(10) 保存网页。

(11) 检验效果。

从浏览器查看网页，并在 Display Mode 下拉列表框中选择 Edit 选项，然后再到各个小窗口中单击“▾”图标，弹出的选项中除原来的【最小化】、【关闭】以外增加了【编辑】项。如果单击【编辑】项，将弹出 EditorZone 中两个控件，利用它们可以对显示的控件进行编辑。例如，将【我的链接】从 SidebarZone 转移到 MainZone 中，如图 22.7 所示。还可以改变外观的标题等。





图 22.6　一个可编辑的主页



图 22.7　改变布局后的情况

22.5.3　运行中增添 WebPart 控件

可以在程序运行中由客户给网页增添新的 WebPart 控件。但是这些控件必须是 Web Part 中的 catalog 控件，而且必须事先放入 CatalogZone 中。目前 ASP.NET 3.5 只提供了两个控件给客户选用，它们是上载文件控件(FileUpload)和日历控件(Calendar)。

下面介绍增添 WebPart 控件的设置方法。

(1) 打开 WebPartDemo.aspx 网页，并转向【设计】视图。

(2) 从工具箱中将 CatalogZone 控件拖到右边列的 EditorZone 控件的下面。由于这两种控件不会同时显示，因此放在同一个 Zone 中不会互相干扰。

(3) 给 CatalogZone 控件的 HeaderText 设置属性。

(4) 从工具箱中将 DeclarativeCatalogPart 控件拖入 CatalogZone 控件中。

(5) 单击 DeclarativeCatalogPart 控件右上方的【编辑模板】项，在弹出的菜单中选择 Edit Templates 命令。

(6) 将工具箱中的 FileUpload 控件和 Calendar 控件拖入到 DeclarativeCatalogPart 控件的模板中。

(7) 转到【源】视图，查看一下<asp:catalogzone>元素的代码，该标记下的 DeclarativeCatalogPart 控件应该将<webpartstemplate>元素以及刚放入的两个服务器控件包括在内。

(8) 给两个控件增加 title 属性。尽管 title 并不是这两个服务器控件的属性，但是在运行中，当客户将这些控件从 Catalog 增加到 WebPartZone 中时，这些控件将被包装在 GenericWebPart 控件中。此时这些控件如同 WebPart 控件，将显示出它们的 title。

此时的代码如下。

```
<asp:CatalogZone ID="CatalogZone1" runat="server">
  <ZoneTemplate>
    <asp:DeclarativeCatalogPart              ID="DeclarativeCatalogPart1"
runat="server">
    <WebPartsTemplate>
      <asp:FileUpload  ID="FileUpload1"  runat="server"  title=" 文 件 上 传
"/><br />
      <asp:Calendar ID="Calendar1" runat="server" title="日历显示">
</asp:Calendar>
    </WebPartsTemplate>
</asp:DeclarativeCatalogPart>
</ZoneTemplate>
</asp:CatalogZone>
```

(9) 保存文件。

(10) 检查效果。

① 打开浏览器查看网页。

② 在 Display Mode 的下拉列表框中选择 Catalog 项。

③ 在目录区选择需要加入的控件(本例中选择【日历显示】)。

④ 在【添加到】下拉列表框中选择放置的区域(本例中选择 SideBar)。

最后显示的结果如图 22.8 所示。



图 22.8　网页中增添控件的情况

22.6　小　　结

网站的个性化服务是一个发展方向，也是一个比较复杂的技术问题，目前正处于新的发展阶段，一些问题还没有完全成熟。

ASP.NET 3.5 对网站的个性化服务提供了强有力的支持。这些支持包括客户的识别和跟踪、对客户关心数据的保存和允许客户定制主页三个方面，分别用 Membership、profile 和 WebPart 来实现。ASP.NET 3.5 对这些工作都实现了自动化，这是对个性化设计的最大支持。设计者只需要在这个基础上做一些设置，例如，为客户分配角色、权限，设置对匿名客户的识别，定义客户关心的数据(Profile)，在个性化网页上部署多重控件提供客户选择、编辑等，就能在一定程度上实现个性化服务的要求。

22.7　习　　题

1. 填空题

(1) 在 ASP.NET 3.5 中，系统对个性化服务提供了三方面的支持。这些支持如下。

Membership: ______________________。

profile: ______________________。

WebPart: ______________________。

(2) 为了识别匿名客户需在 Web.config 文件中作如下设置。

```
<system.web>
      <anonymousIdentification __________/>
</system.web>
```

(3) 如果在 Web.config 文件中定义了一个 profile。

```
<profile>
  <property name="Name" />
</profile>
```

这个 profile 的名字是_____，类型是____________。

(4) 为了将定义的 profile 提供给匿名客户使用，在定义的后面还要添加如下属性。

```
<profile>
        <properties>
           <add name=" PhoneCode " type="string" _________________/>
        </properties>
        </profile>
```

2. 判断题

(1) profile 是一组为特定客户定义且可以保存的数据。　(　)

(2) 放入 Zone 控件中的 WebPart 控件不受限制。　(　)

(3) 能够编辑的 WebPart 类只能放入 EditorZone 中。　(　)

3. 简答题

(1) 简述在个性化网页中保留并显示自己关心数据的方法。

(2) WebPartManager 控件起什么作用？

(3) WebPartZone 控件的作用是什么？它与 Web Part 之间是什么关系？

4. 操作题

(1) 将个人喜爱的多个网址保存在自己的个性化网页中，每当打开网站时立即显示这些网址。

(2) 利用 WebPart 控件创建一个简单的个性化网页。

第五部分

ASP.NET Ajax

Ajax 是浏览器与服务器通信技术的革新，2005 年年初被提出后就引起了广泛的注意，特别在 Google、Microsoft 两大软件巨人的大力倡导下，得到了很大的发展。目前已经广泛应用于各种类型的 Web 应用中。本部分将先介绍 Ajax 的基本原理，然后重点介绍在 ASP.NET 中的应用。具体内容分为三章：

- Ajax 原理。
- ASP.NET Ajax 技术。
- Ajax 工具箱。

要求在掌握 Ajax 基本原理的基础上，重点掌握 ASP.NET 服务器端的 Ajax 技术，能够设计 ASP.NET Ajax 网站，或者根据需要对传统的 Web 网站进行 Ajax 升级。

第 23 章　Ajax 原理

为了更好地理解 Ajax 原理，本章将通过与传统通信方式进行比较来说明 Ajax 的特点和优点，然后通过一个示例来说明 Ajax 的实现过程，最后再介绍一种 XML 的“瘦身”语言 JSON。

23.1　概　　述

Ajax 是 Asynchronous(异步)+JavaScript + XML 的简写，它是对浏览器与服务器之间通信方式的革新。为了理解 Ajax 原理，先回顾一下传统的浏览器与服务器之间的通信过程。

23.1.1　传统的浏览器与服务器的通信过程

运行 ASPX 网页时，传统的浏览器与服务器之间的通信过程是这样的。

(1)　客户通过 URL 向服务器申请网页。

(2)　在服务器端生成网页后发送给浏览器。

(3)　浏览器下载并编译 HTML 后显示在网页上。

(4)　在浏览器端发生了事件，将信息回访给服务器处理。

(5)　当服务器处理完事件后生成新网页，再返回浏览器。

(6)　浏览器下载后刷新并显示新的网页。

其过程如图 23.1 所示。

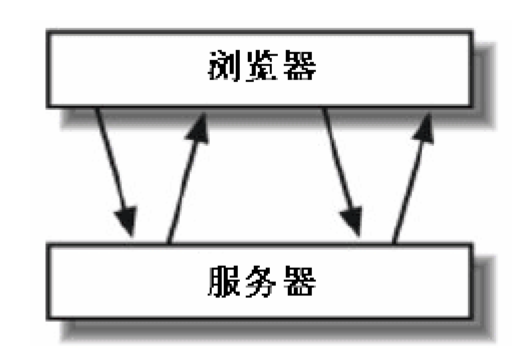


图 23.1　传统的浏览器与服务器的通信过程

由于传统的通信采用同步方式，当向服务器发送信息以后，要等待服务器处理完毕返回后，浏览器才能继续执行后续的操作。其过程如图 23.2 所示。

传统通信的过程表明，第一次访问网页与发生事件后回访的过程基本相同，每次都需要对网页的整体进行传送和刷新。其结果不仅通信量大，而且由于网页整体刷新带来的闪烁，加上同步通信带来的间断都影响了系统的运行效率，降低了客户的体验。

23.1.2　Ajax 模式下的通信过程

为了改善传统的通信过程，在浏览器端增添一个中间层——Ajax 引擎。数据传输时浏

览器与服务器不再直接打交道，而首先要将信息传送到 Ajax 引擎。Ajax 引擎实际上是一组 JavaScript 对象或函数，用来在浏览器中对元素定位，并作一些初步的处理工作，然后再送出部分数据。

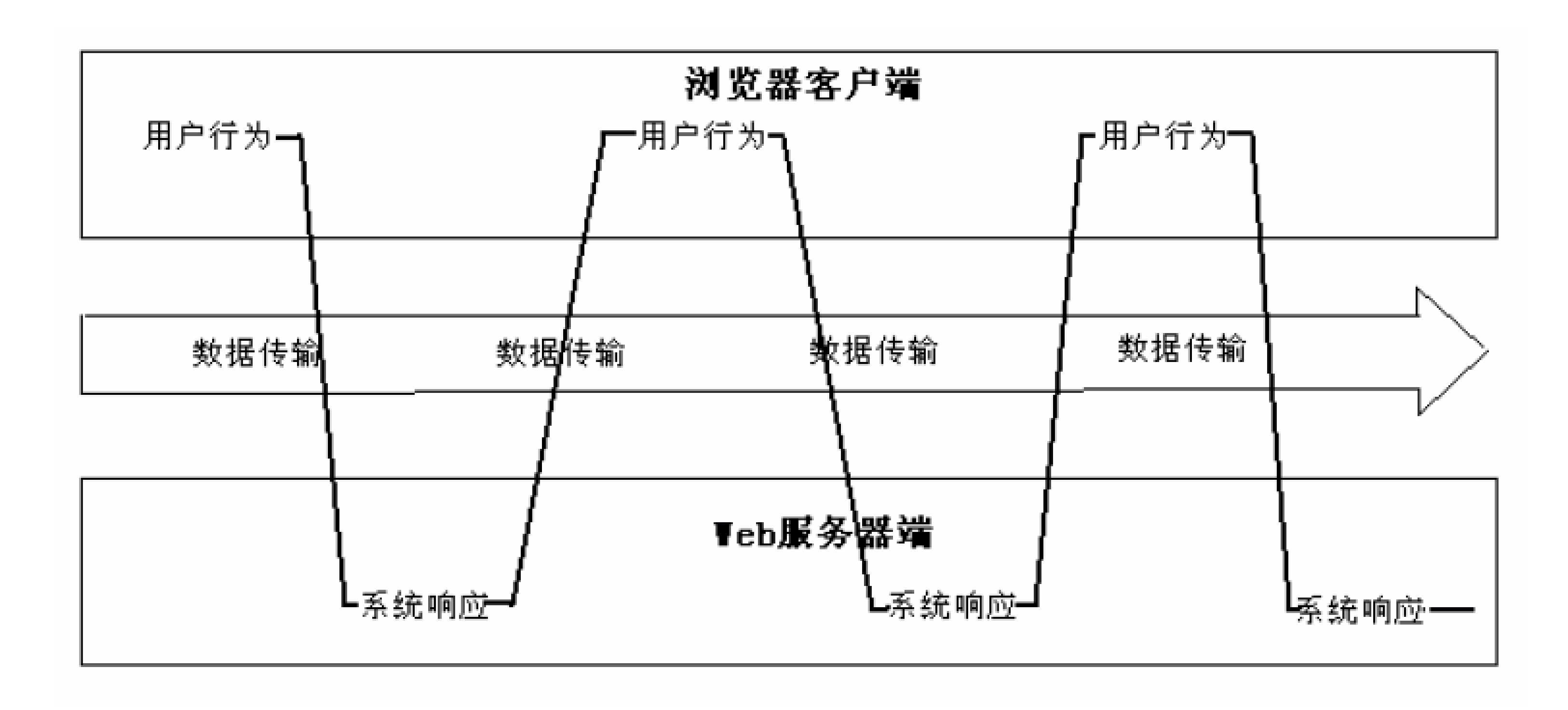


图 23.2　信息的传输过程

Ajax 模式下浏览器与服务器之间的通信如图 23.3 所示。

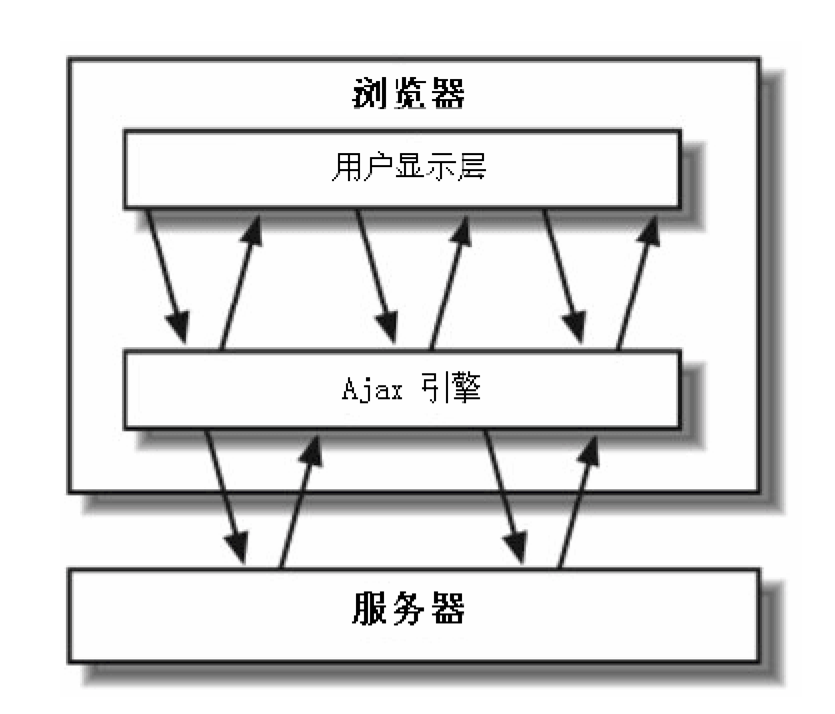


图 23.3　Ajax 模式下浏览器与服务器之间的通信过程

另外，Ajax 的通信方式由同步改为异步(Asynchronous)。即当浏览器向 Ajax 引擎发送信息后不等处理完毕即返回继续执行后续代码。异步通信的过程如图 23.4 所示。

在 Ajax 模式中，第一次访问服务器与后续的回访是不同的。第一次访问网页时，服务器不仅要将网页本身，而且要将浏览器需要的 JavaScript 代码(一些增强浏览器功能的代码)，一并发送到浏览器。因此第一次传送的信息量比传统方式要多一些，但后续的回访中只交换一些改变了的数据(不是整个网页)，因此通信量将大大减少。

23.1.3　信息流通量的比较

现在将上述两种通信方式的通信流量做一些比较。

- 用传统方法进行交互时，每次交互都是整体网页的传送和更新，因此信息流通量都差不多，如图 23.5(a)所示。
- 用 Ajax 模式进行通信时后续访问的通信量大大低于第一次访问时的通信量，如

图 23.5(b)所示。

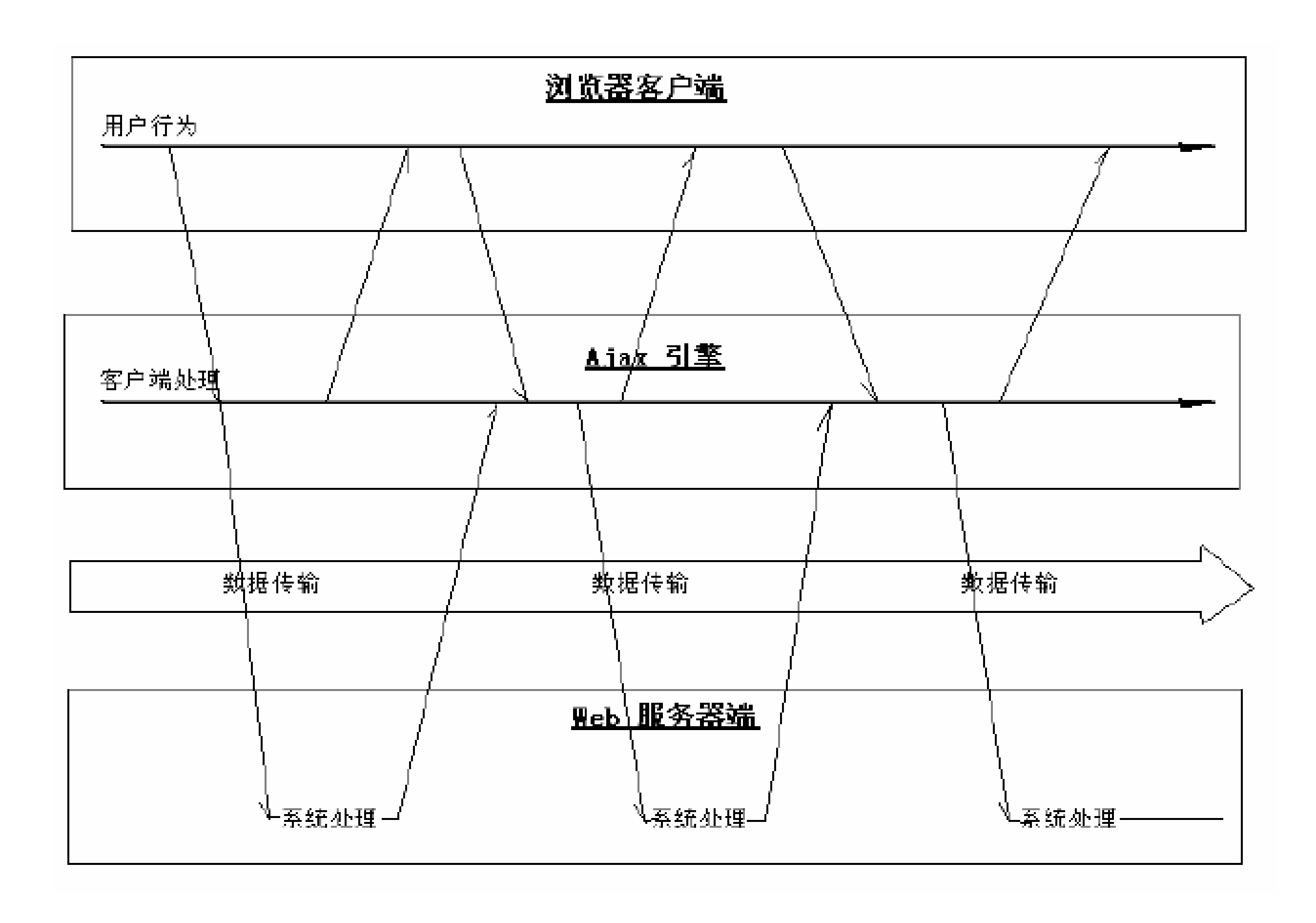


图 23.4　Ajax 异步通信的过程

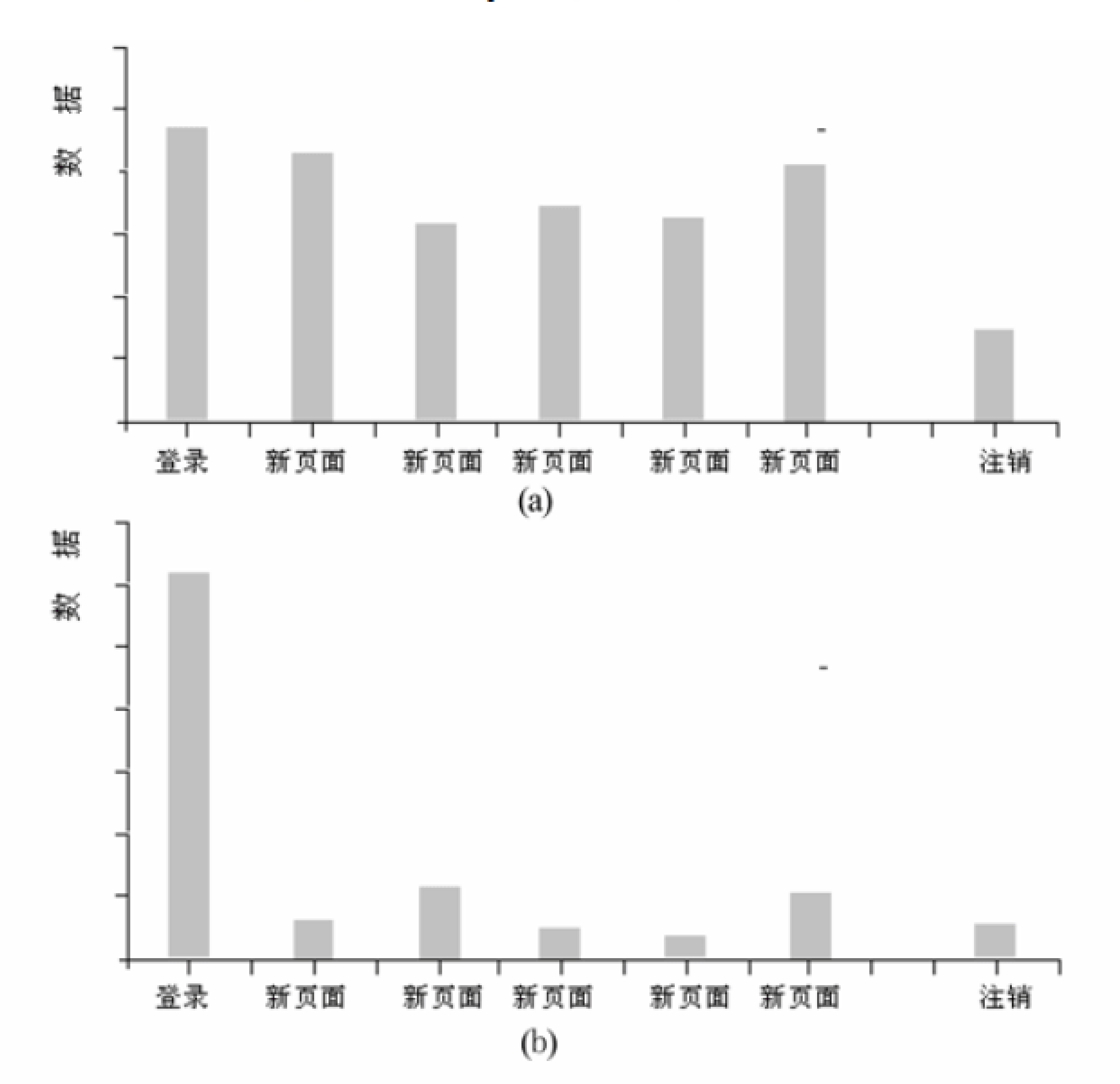


图 23.5　两种通信模式的信息流通量

- 两种通信模式的通信总量的比较如图 23.6 所示。

  图中实线代表传统通信方式时的情况，其通信总量将随回访次数的增加而大幅度增加；虚线代表用 Ajax 进行通信时的通信总量，它表明当回访同一网页时，总通信量的增加并不显著。

概括地说，Ajax 对传统通信的方式作了两方面的改进。

- 用局部数据传送和刷新，代替网页的整体传送和刷新。
- 用异步通信代替同步通信。

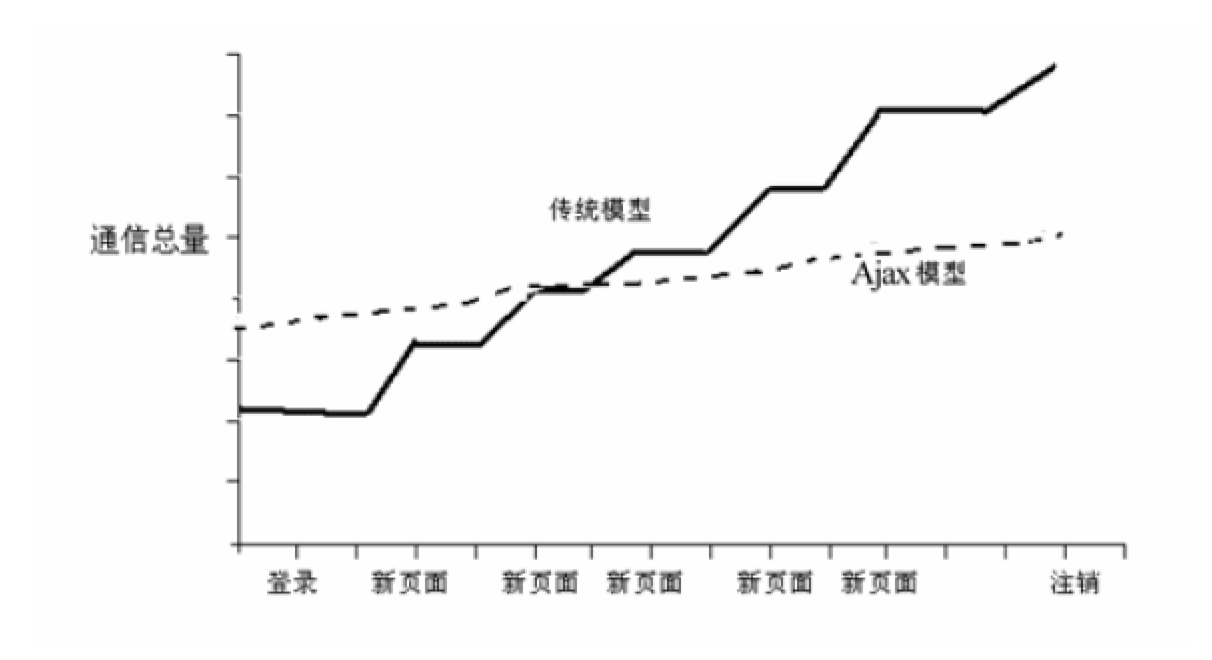


图 23.6　两种通信方式通信总量的比较

Ajax 模式使 Web 的运行方式更接近于桌面系统的运行方式。和传统的通信模式比较，Ajax 模式不仅减少了信息的总流通量，而且客户与浏览器之间的交互变得更加灵敏，客户体验得到了明显的改善。

23.2　Ajax 的组成

Ajax 并不是单一的技术，而是几种已经比较成熟技术的组合。包括如下几种技术。

- DOM：Web 网页由 DOM 网页元素组成，这些元素可以通过 JavaScript 脚本进行查询和修改。
- CSS：CSS 为 Web 页面元素提供了一种可重用的定义方法。在 Ajax 的应用中客户界面的样式可以通过 CSS 独立进行修改。
- JavaScript：JavaScript 是通用的脚本语言。通过 JavaScript 解释器可以执行浏览器的内建功能。Ajax 程序实际上是用 JavaScript 语言编写的。
- XMLHttpRequest 类：XMLHttpRequest 是用于通信的类，它允许用异步(也可以用同步)方式与服务器进行交互，交互的数据格式通常是 XML 或 JSON(下面讲述)。

以上 4 个部分的关系如图 23.7 所示。

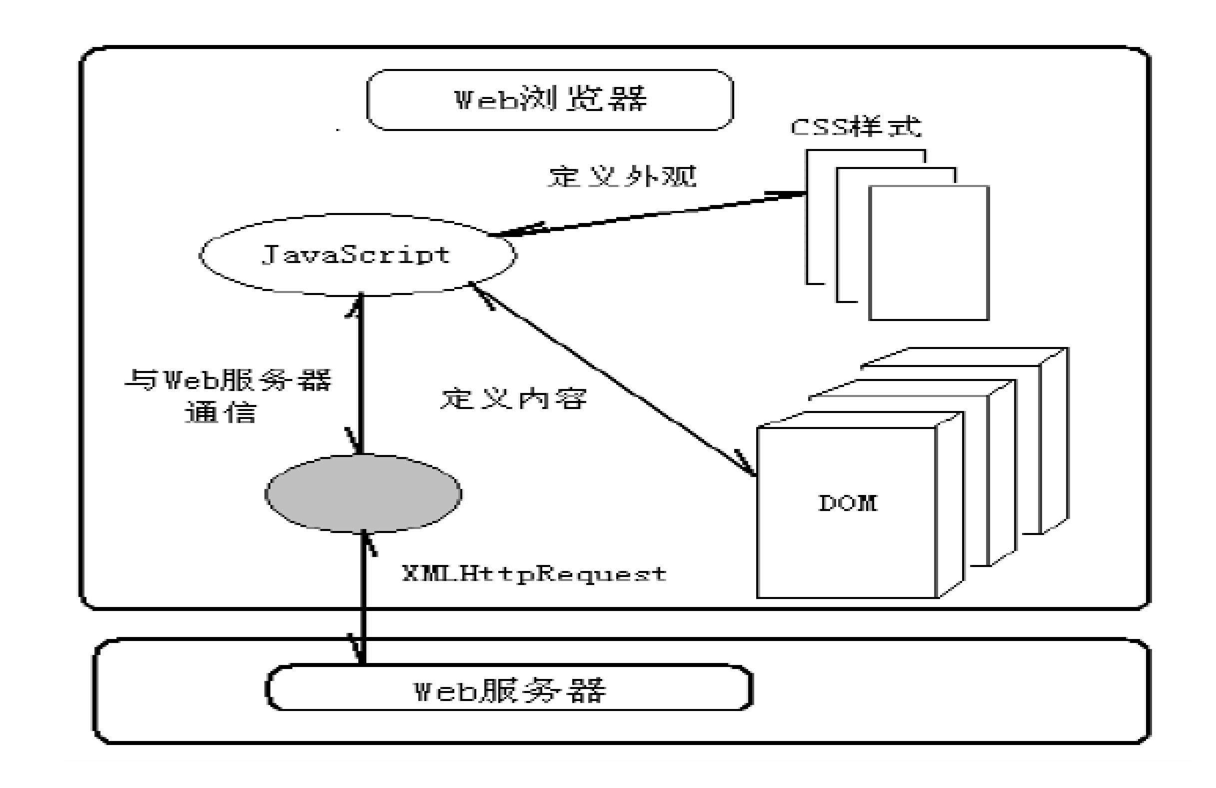


图 23.7　Ajax 的组成

上述技术在 DHTML 和 jQuery 中都用到过，现在只是对这些技术重新进行组合并加

以延伸和增强，从而发挥出新的作用。

Ajax 的组成说明，应用 Ajax 技术并不需要增加什么新硬件，也不需要下载什么大型软件。世界上几乎所有的计算机(包括桌面系统、手提电脑等)都已经为 Ajax 的应用准备好了必要的物质条件。

23.3 Ajax 中的几个关键语句

下面是几个与 Ajax 技术相关的关键语句。

23.3.1 通信类的兼容语句

从前面的分析可以看出，XMLHttpRequest 通信类在 Ajax 中起到了关键的作用，正是通过它发送异步请求、接收响应以及执行回调等项任务。XMLHttpRequest 最初是作为 IE 5 Active 组件引入的，后来各大浏览器厂商都提供了类似的产品。它们的原理相同，结构也只有很小的区别，但各厂商对该类的取名却不同。IE 浏览器使用的是 ActiveXObject 类。其他类型的浏览器，如 Mozilla Firefox、Safari 和 Opera 等使用的名称是 XMLHttpRequest 类。因此不同类型的浏览器生成该类对象的语句也各不相同。

对于 IE 浏览器来说，生成通信对象的语句是

```
var oXmlHttp = new
                ActiveXObject("Microsoft.XMLHTTP");
```

对于其他类型的浏览器来说，生成通信对象的语句是

```
var oXmlHttp = new XMLHttpRequest();
```

为了使 Ajax 技术能够应用于不同类型的浏览器中，首先要解决通信类在各浏览器之间的兼容问题，还要考虑不同版本的应用问题(微软的 IE 已经开发了多种版本)。

综合上述因素，生成通信对象的语句是

```
function createXMLHttp(){
  // 如果已经定义了 XMLHttpRequest
  if (typeof XMLHttpRequest != "undefined") {
      return new XMLHttpRequest();
    }
else if(window.ActiyeXObject){  //定义 IE 的最新版本
var aversion = ["MSXML3.XMLHttp.7.0"," MSXML3.XMLHttp.6.0",
              "MSXML5.XMLHttp","Microsoft.XMLHttp"];
    // 为了选择使用最新版本
for (var i=0;i < aversion.length; i++) {
try {
       var oXmlHttp = new ActiveXObject(aversion[i]);
       return oXmlHttp;
       }catch (oError) {
      // Do nothing
    }
  }
}
  Throw new Error("XMLHttp object could not be created。");
}
```

23.3.2　浏览器中元素定位语句

在浏览器中，用 JavaScript 对元素定位的语句有以下两种。

- 根据元素 ID 定位：

```
document.getElementById("someID");
```

- 根据元素名：

```
document.getElementsByTagName("someTag");
```

23.3.3　异步通信的语句

```
xhr = new ActiveXObject("Microsoft.XMLHTTP");
    xhr.open("Get",url ,true);
// 参数中的 true 代表启用异步通信，如果使用 false 则为同步通信
    xhr.send(null);    //发送信息
    xhr.onreadystatechange=functionName;
// functionName 代表返回结果后执行的函数名
```

23.4　Ajax 异步通信示例

下面通过一个简单的示例来说明 Ajax 的异步通信过程。本示例的功能是，根据客户输入的邮政编码找到并填入客户的具体地址。

首先建立数据库及数据表。假定邮政代码的简表如图 23.8 所示，注意地址名之间用句号分隔，例如“湖南省.长沙市.宁乡县.”，如果有缺项时，要用句号填补，如“北京市...”等。

bh	address	mailcode
1	北京市...	100000
2	上海市...	200000
3	江苏省.南京市..	210000
6	湖南省.长沙市..	410000
7	湖南省.长沙市.长沙县.	410100
8	湖南省.长沙市.望城县.	410200
9	湖南省.长沙市.浏阳市.	410300
10	湖南省.长沙市.宁乡县.	410600

图 23.8　各省、市、县邮政编码简表

23.4.1　浏览器端的设置

在 HTML 网页中，用浏览器端控件设置的显示界面如图 23.9 所示。

下面的代码中包括两个函数。

- function getZipData(code)用来发出异步通信请求。
- function processZipData()通信完成后对结果进行处理。

邮政编码：
省或直辖市：
市：
县：
其他信息：
submit

图 23.9　HTML 网页显示界面

```
<script language="javascript" type="text/javascript">
    var xhr;
      function getZipData(code)
   {
    if(window.ActiveXObject)  //这里只生成微软的通信对象
    {
      xhr = new ActiveXObject("Microsoft.XMLHTTP");
     }
     else
     {
      xhr = new XMLHttpRequest();
      }
url="Default.aspx? addcode="+document.getElementById("Text1").value;
                                //给 DOM 中的元素定位
     xhr.open("Get",url ,true); //以 Get 方式与 url 之间进行异步通信
     xhr.send(null);            //发送信息
     xhr.onreadystatechange=processZipData;
       // 指定 processZipData 为返回时执行的函数
     }
   function processZipData()
   {
   if(xhr.readyState == 4)           //判断异步通信是否完成
    {
    var data = xhr.responseText;      //取出回调信息
    var cityState = data.split('.'); //分解回调信息
    document.getElementById("Text2").value = cityState[0];    //显示信息
    document.getElementById("Text3").value = cityState[1];
    document.getElementById("Text4").value = cityState[2];
    }
}
function Text1_onblur() {  //事件代码
   getZipData(this.value);
}
</script>
```

其中几条语句的作用如下。

- Xhr.open("Get",url ,true)语句的作用是启动异步通信。
- if(xhr.readyState == 4)的作用是判断通信是否完成。
- 语句 document.getElementById("…").value 代表到控件中取值。
- var cityState = data.split('.');语句中的 split('.')代表将 cityState 字符串以“.”为界分解成字符串数组，如 cityState[0]、cityState[1]、cityState[2]等。

23.4.2　服务器端的设置

浏览器调用与服务器响应是两个独立的过程。浏览器只要将信息传出，不需等待即可开始其他工作，待服务完成后再对结果进行处理。

服务器可以是 Java/J2EE，也可以是.NET 系统，主要任务是根据传来的邮政编码，在数据库中查出对应的地址，再用 Response.Write()方法送回浏览器。

这里采用了 ASP.NET 3.5 的三层架构，利用存储过程从数据库中查出具体地址。

为此，先写一个存储过程，代码如图 23.10 所示。

另外在数据集中增添一个方法 getaddress(@mailcode)，如图 23.11 所示。

```
ALTER PROCEDURE dbo.getaddress
  (
 @mailcode nvarchar(20)
 )
 AS
 select address from mailcode where mailcode=@mailcode
 RETURN
```

图 23.10　存储过程代码

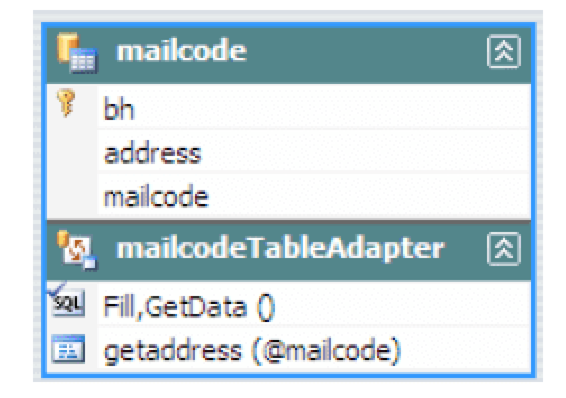


图 23.11　在数据集中增添方法

并在.aspx 网页中编写如下代码。

```
public partial class _Default : System.Web.UI.Page
{
  DataSet1TableAdapters.mailcodeTableAdapter tt;
  protected void Page_Load(object sender, EventArgs e)
  {
   tt = new DataSet1TableAdapters.mailcodeTableAdapter();

   if (Request.QueryString["addcode"] != null)
    {
      string add = (string)tt.getaddress(Request.QueryString["addcode"]);
      System.Threading.Thread.Sleep(10000); //为了演示而附加的代码，延时 10 秒
      Response.Write(add);
  }
}
}
```

运行上述程序后，只要客户在网页中输入邮政编码，就可以从服务器返回的信息中分解出相应的地址(省、市、县)。为了演示异步通信的效果，在上述服务器端代码中增加了“延时”部分。演示时当从浏览器填完邮政代码并提交以后，不必等待服务器处理，立即可以继续进行后续的工作(例如继续填写【其他信息】栏)。

23.5　JSON 语言

随着 Web 的广泛流行，XML 已经实际上成为数据传输的标准。但是 XML 不是没有缺点，也不是没有批评者。例如，有些人认为，由于 XML 语句的前后必须包括标签，从而大大增加了数据的冗余，即使是一个简单的表单也往往需要传输大量的信息。由于这种

原因，有人为此另外开发了 XML 的压缩格式，甚至是全新的 XML 表单，诸如二进制 XML(Binary XML)等。这些解决方案都可以看成对 XML 的扩展或补充。一位资深的软件工程师 Douglas Crockford 专门开发了一个内建于 JavaScript 的数据格式，称为 JavaScript 的对象表示(JSON)，常用于 Ajax 的通信中，以进一步减少通信流量。

23.5.1 什么是 JSON

JSON 是 JavaScript Object Notation 的缩写，它是一种轻量级的数据格式，目的在于简化 XML 的表达方式。它基于 JavaScript 语法的子集，常用数组和对象来表示。由于它使用的是 JavaScript 语法，因此 JSON 可以包含在 JavaScript 文件中，对其访问无须通过 XML 的语言来进行额外的解析。不过使用 JSON 需要理解 JavaScript 中数组及对象特殊的语法结构。

23.5.2 数组字面量

数组字面量是用一对方括号括起来的，用逗号隔开的 JavaScript 值，包括字符串、数字、布尔值或 null。例如：

```
var aName=["北京","湖南","长沙"];
```

然后可以用数组的名称和方括号来访问该数组的值。例如：

```
alert(aName[0]);  //输出"北京"
alert(aName[1]);  //输出"湖南"
alert(aName[2]);  //输出"长沙"
```

注意：下标序号从 0 开始。

由于 JavaScript 中的数组是无类型的，因此可以存入任何类型的值。例如：

```
var aValues=["string", 24 , true , null];
```

如果不想使用字面量表示法来定义数组，也可以用 Array 的构造函数。例如：

```
var aValues = new Array("string", 24,true, null);
```

虽然在 JavaScript 中，上述两种方法都是可行的，但在 JSON 中只能用字面量。

23.5.3 对象字面量

对象字面量用来存储“名称－值”对的信息，最终将创建一个对象。对象字面量通过大括号({ })来定义。在大括号中可以放置任意数量的“名称－值”对，定义的格式是“字符串值”。除最后一行外，每个“名称－值”对后必须有一个逗号。例如：

```
var oCar = {
    "color" : "red" ,
    "doors" : 4 ,
    "paidFor" : true
};
```

这段代码创建了一个对象，其中包括 3 个名字分别是 color、doors 和 paidFor 的属

性。每个属性的值都不一样，可以通过以下语句来访问它们。

```
alert(oCar.color);    //输出 red
alert(oCar.doors);    //输出 4
alert(oCar.paidFor);  //输出 true
```

也可以使用方括号，并将属性的名称作为字符串值放入其中，来访问属性。如：

```
alert(oCar["color"]);    //输出 red
alert(oCar["doors"]);    //输出 4
alert(oCar["paidFor"]);  //输出 true
```

在这里可以看出，使用对象字面量要比对象的构造函数代码量要少一些。

23.5.4 混合字面量

可以混用对象和数组字面量，来创建一个对象数组，或者一个包含数组的对象。例如：

```
     var aCars = {
        {
          "color" : "red",
          "doors" : 2 ,
          "paidFor" : true
        }
        {
          "color" : "blue",
          "doors" : 4 ,
          "paidFor" : true
        }
        {
          "color" : "white",
          "doors" : 2,
          "paidFor" : false
        }
  };
```

这段代码定义了一个数组，包括三个对象，每个对象都包括 color、doors 和 paidFor 三个属性。可以通过组合方括号和“.”符号来访问数组的信息。例如：

```
alert(aCars[1].doors);  //输出 4
```

同样，也可以在对象字面量中定义一个数组。如：

```
     var oCarInfo = {
        "availableColors" : ["red", "white", "blue"] ,
        "availableDoors" : [2, 4]
        }
```

这段代码定义了另一个对象名为 oCarInfo，它包括 availableColors 和 availableDoors 两个属性。这两个属性都是数组，分别包含字符串和数字。要访问它们，只需将方括号和“.”符号顺序调用即可。因此，要调用第二个可用的颜色值时，可以采用以下语句。

```
alert(oCarInfo.availableColors[1]);
```

23.5.5 JSON 语法

JSON 只是用来表示数据，它的语法中除了混合对象和数组表示来存储的数据以外，没有其他更多内容。在 JSON 中没有变量、赋值、判断等概念。因此，JSON 的原始代码与最后的代码是相似的。最后的代码如下。

```
{
 "availableColors" : ["red","white","blue"] ,
 "availableDoors" : [2,4]
}
```

JSON 的语法结构如图 23.12 所示。

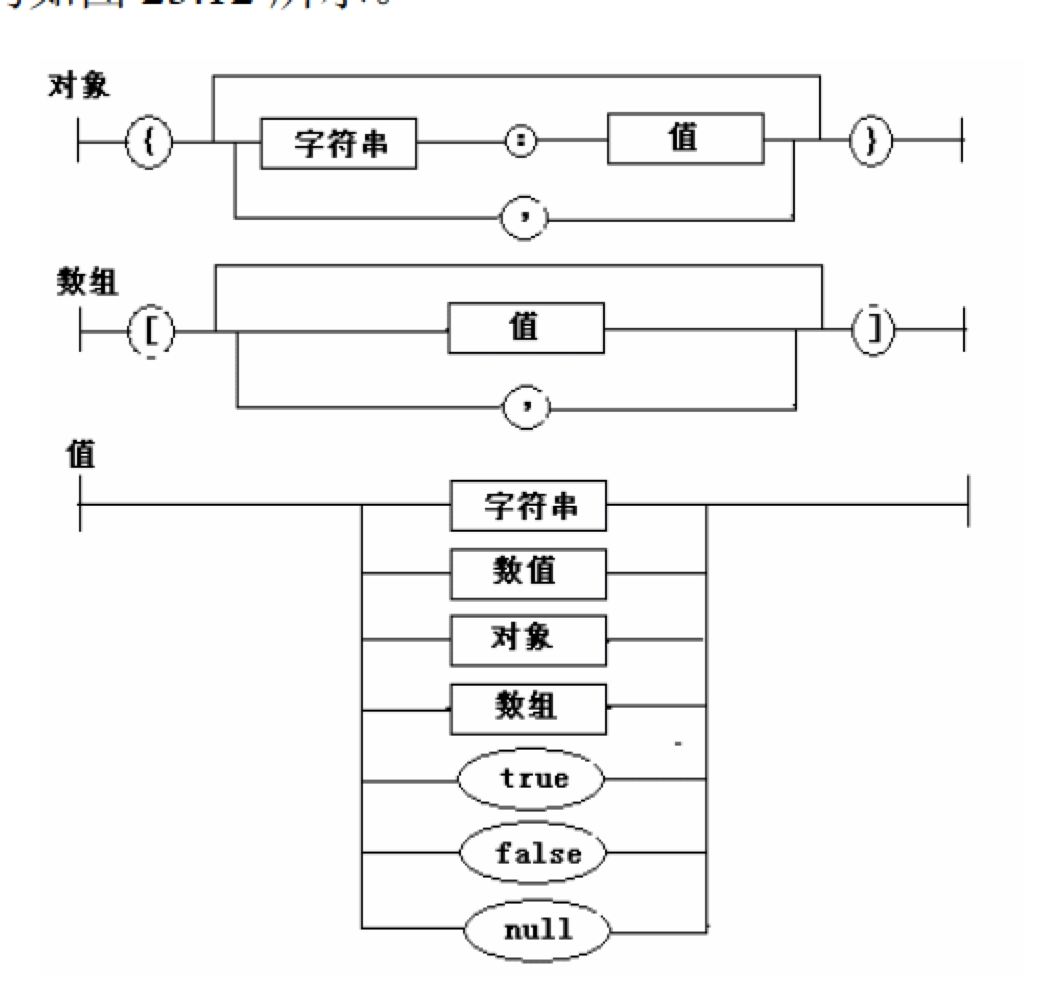


图 23.12 JSON 语法结构图

和原始代码相比，oCarInfo 变量去掉了，但在大括号中仍然包括冒号和逗号。如果将其通过 HTTP 传送给浏览器，速度将会很快，因为字符的总数很少。假定通过 XMLHttp(或者其他客户端－服务器端通信机制)这些数据并存到 sJSON 变量中，那么这时将拥有一个字符串，而非一个对象，特别是包含两个数组的对象，必须使用 JavaScript 中的 eval()函数来实现。其方法如下：

```
var oCarInfo = eval("(" + sJSON + ")");
```

在 JavaScript 通信中使用 JSON 作为数据格式的好处很明显：可以立即获得数据的值，因此可以更快地访问其中包括的数据。

23.5.6 JSON 与 XML 比较

下面是用 XML 来描述一个班中三名学生的信息。

```
<classinfo>
  <students>
    <student>
      <name>张三</name>
```

```
      <average>95</average>
      <age>18</age>
    </student>
    <student>
      <name>李四</name>
      <average>83.8</average>
      <age>20</age>
    </student>
    <student>
      <name>王五</name>
      <average>66</average>
      <age>21</age>
    </student>
    </students>
</classinfo>
```

实际上对于这些数据来说，我们感兴趣的只有与学生相关的信息。文件中前后的标签如<students>和<classinfo>等只是 XML 语法的需要。而且每个信息都在前后重复使用了两次。这些都增加了通信的流量。

下面是用 JSON 改写后的描述。

```
{ "classinfo" :
  {
   "students" : {
   {
     "name" : "张三",
     "average" : 95,
     "age" : 18
   }
   {
     "name" : "李四",
     "average" : 83.8,
     "age" : 20
   }
   {
     "name" : "王五",
     "average" : 66,
     "age" : 21
   }
   }
 }
}
```

比较上面两种表示形式可以看出，同样的内容用 JSON 表示比原 XML 文档少将近 100B(约占 50%)。这是一个最简单的情况。对于一些比较复杂的文档来说，节约的数量会更多。因此创始人 Crockford 将 JSON 称为对 XML 的“减肥方案”。

然而任何事物都是一分为二的。XML 多一些标签，提高了文件的可读性，人们可以更准确地理解文档所表达的内容。JSON 的符号更加简化，用肉眼观察有时可能会增加一点难度。但对于数据交换来说，交换的格式都是用机器识别，从不用肉眼观察。因此两者各有自己的优缺点，互不排斥，分别适用于不同的场合。

23.6 小　结

Ajax 技术是对浏览器与服务器之间通信方式的革新。在传统的通信方式中对网页的访问都是整体传送和刷新，加上用同步方式通信，不仅通信流量大，而且使客户体验下降。现在 Ajax 用增加 Ajax 引擎的办法增强浏览器的功能。在 Ajax 模式下浏览器与服务器不再直接打交道，而是首先通过 Ajax 引擎。Ajax 引擎实际上就是一组 JavaScript 的类和函数，用来给浏览器中的元素进行定位并作一些初步处理，然后将部分数据传送到服务器去。从服务器返回的数据也首先交给 Ajax 引擎，再分送到浏览器的元素。用局部数据的传送和刷新代替了整体网页的传送和刷新；用异步通信代替同步通信是 Ajax 改善传统方式的两大措施，实现了这些措施就能大大改善客户体验。为此系统需要解决通信类的兼容问题、异步通信中出现的各种问题。

JSON 是 XML 的减肥方案，在 Ajax 中利用它来传输数据将进一步减少信息的流通量。

23.7 习　题

1. 填空题

(1) Ajax 是___________________的英文缩写。

(2) 在 Ajax 模式下的通信中，浏览器与服务器不再直接打交道，而是先将信息传送给_________。

(3) 在 IE 中异步通信类的名字是_________。

(4) 在 Mozilla Firefox 中异步通信类的名字是_________。

2. 选择题

(1) Ajax 引擎实际上是一组________。

A. 数学表达式

B. 指向 JavaScript 的路径

C. JavaScript 的对象和函数

D. 字符串

(2) Ajax 引擎的作用是________。

A. 定位元素　　B. 异步通信的类　　C. 初步处理数据　　D. A+C

(3) 下列语句中带下划线的部分代表________。

```
Xhr = new ActiveXObject("Microsoft.XMLHTTP");
Xhr.open("Get",url,true);
```

A. 打开数据库　　B. 启动异步通信

C. 连接数据库　　D. 启动同步通信

(4) JSON 语言的目的是________。

A. 简化 HTML 的表达方式　　B. 简化 JavaScript 的表达方式
C. 简化 CSS 的表达方式　　D. 简化 XML 的表达方式

3. 判断题

(1) Ajax 技术适用于对不同网页的访问。　( )
(2) Ajax 技术适用于对同一网页的多次访问。　( )

4. 简答题

(1) 同步通信与异步通信的区别是什么？
(2) Ajax 从哪些方面改善了传统的通信方式？
(3) 对比分析传统通信方式与 Ajax 模式下通信方式总信息流通量的变化。
(4) 不同类型的浏览器异步通信类的名称不同，系统是如何解决通信类兼容问题的？

5. 操作题

模仿 23.4 节的示例创建一个用 Ajax 异步通信方法访问学生成绩的应用程序。

第 24 章　ASP.NET Ajax 技术

上一章已经讲述了 Ajax 的基本原理，如果直接按照 Ajax 的原理进行软件开发难度仍然较大，它要求设计者熟练掌握 JavaScript 语言。现在微软公司在这个基础上推出了 ASP.NET Ajax 技术(以前称为 Atlas)，不仅保持了原 Ajax 的主要优点，还对某些功能进行了扩展，并大大简化了学习和开发的过程。按照微软自己的提法，它的目标不仅要提高客户体验，还要提高开发人员、应用人员以及管理人员的体验。

本章将通过一些示例重点讲解基于服务器的 ASP.NET Ajax 程序开发。具体问题包括:

- ASP.NET Ajax 的特点。
- ASP.NET Ajax 的架构。
- ASP.NET Ajax 控件。
- 服务器端 ASP.NET Ajax 应用示例。
- 给传统网页增添 Ajax 功能。

24.1　ASP.NET Ajax 的特点

ASP.NET Ajax 的特点可以归纳成以下几个方面。

- ASP.NET Ajax 可以与 ASP.NET 无缝地集成。
- 系统提供了脚本库、类库和若干新组件和控件，大大增强了 JavaScript 语言面向对象的特点，又简化了开发过程。
- 系统中不仅开发了基于客户端的 ASP.NET Ajax 应用程序，还允许创建基于服务器的 ASP.NET Ajax 应用程序。系统对两种开发方式都提供了有力的支持。

24.2　ASP.NET Ajax 的架构

微软为 ASP.NET Ajax 提供了服务器端和浏览器端 Ajax 框架两部分，允许用两种方式来开发 Ajax 应用程序。

- 基于服务器端框架开发时，几乎不需要编写 JavaScript 代码，与开发 ASP.NET 的网页相比改变不大，可以与 ASP.NET 的传统程序无缝集成。
- 基于浏览器端框架开发时，能够充分发挥 Ajax 效能。为了降低开发的难度，系统提供了 JavaScript 脚本库，并对 JavaScript 语言增强了很多面向对象的特点(如命名空间、继承等)，从而简化了开发的过程。

ASP.NET Ajax 的架构由两部分组成，其结构如图 24.1 所示。

24.2.1　客户端架构

概括地说，客户端架构中主要包括：

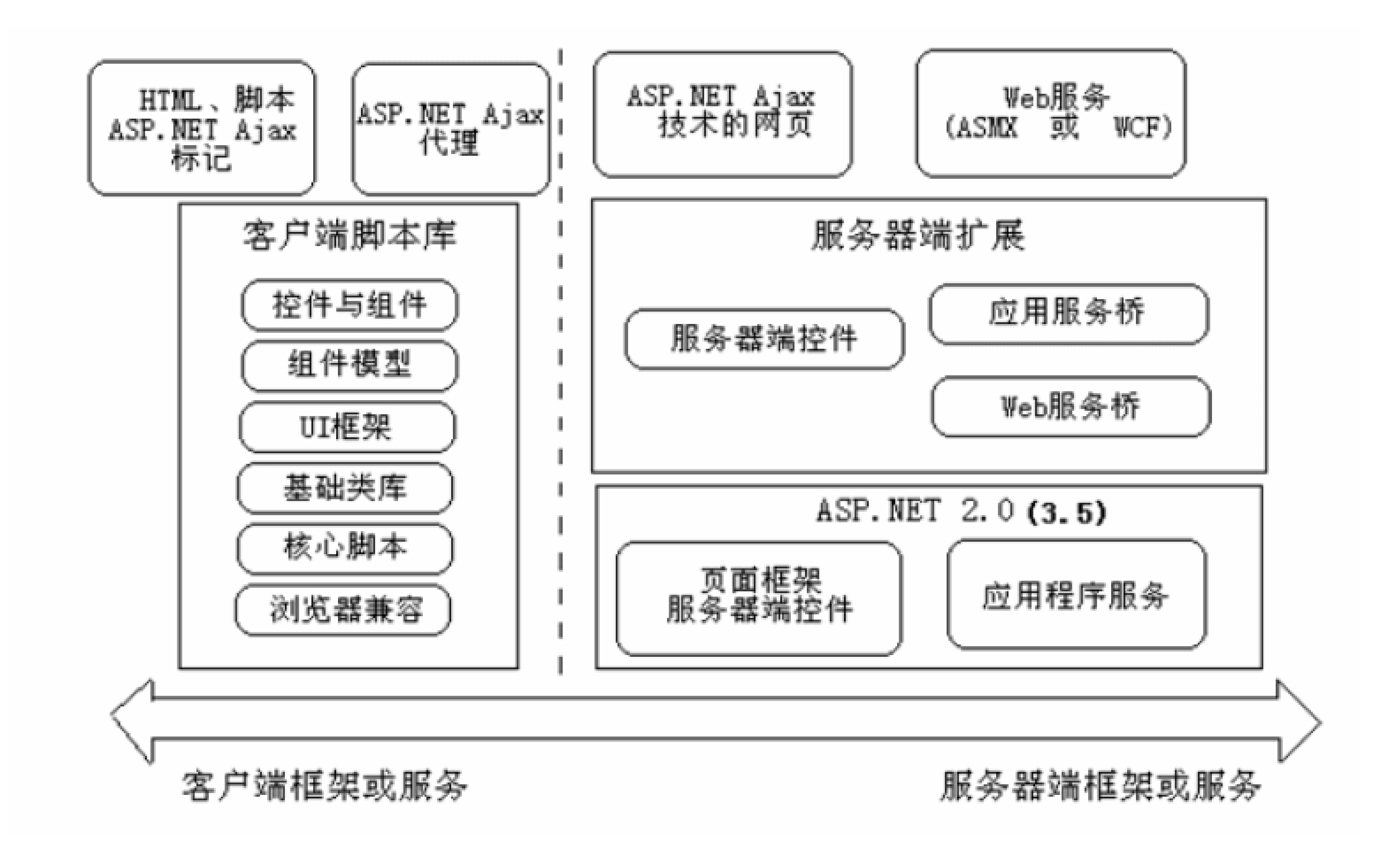


图 24.1　ASP.NET Ajax 框架

- 用来解决浏览器兼容的代码。
- 客户端脚本库。用来帮助开发者高效地为 Web 编写应用程序和实现客户端功能。客户端脚本库具有丰富的组件模型，并允许开发者以面向对象的方式进行脚本编程。脚本库中包括很多常用类，如网络访问、客户界面增强，以及行为、动作和字符串操作等。
- 若干浏览器端控件与组件。

客户端架构的作用很多都体现在 Ajax 工具箱(ToolKit)中，详见第 25 章。

24.2.2　服务器端架构

服务器端架构建立在 ASP.NET 3.5 应用程序的模型之上，既能够实现 Ajax 功能，又能与 Web 服务等最新技术紧密聚合。系统还增加了几个用于实现 Ajax 功能的控件。

24.3　ASP.NET Ajax 控件

为了简化 Ajax 设计，系统提供了 5 个 Ajax 控件。

1. ScriptManager 控件

顾名思义，ScriptManager 是一个对脚本(Script)进行管理(Manager)的控件。所有 ASP.NET Ajax 网页都必须有，而且也只能有一个 ScriptManager 控件，而且此控件必须放在 form 内，所有其他控件的前面。本控件的作用是负责协调哪些 JavaScript 代码需要发送到客户端，并部署和处理这些脚本。

控件中的 EnablePartialRendering 属性指定了用何种方式进行刷新。如果该属性为 true(这是默认设置)，将实现局部刷新；false 时为整个网页刷新。

若在网页中增添“AJAX Web 窗体”或“Ajax 母版页”时，新网页中将会自动增加这一控件。

2. UpdatePanel 控件

UpdatePanel 控件用来限制网页中允许刷新的部分。一个网页中可以放置多个 UpdatePanel 控件，分布在网页的不同区域，通过不同的控件来触发它们进行刷新。

使用 UpdatePanel 控件时，数据交换的情况如图 24.2 所示。

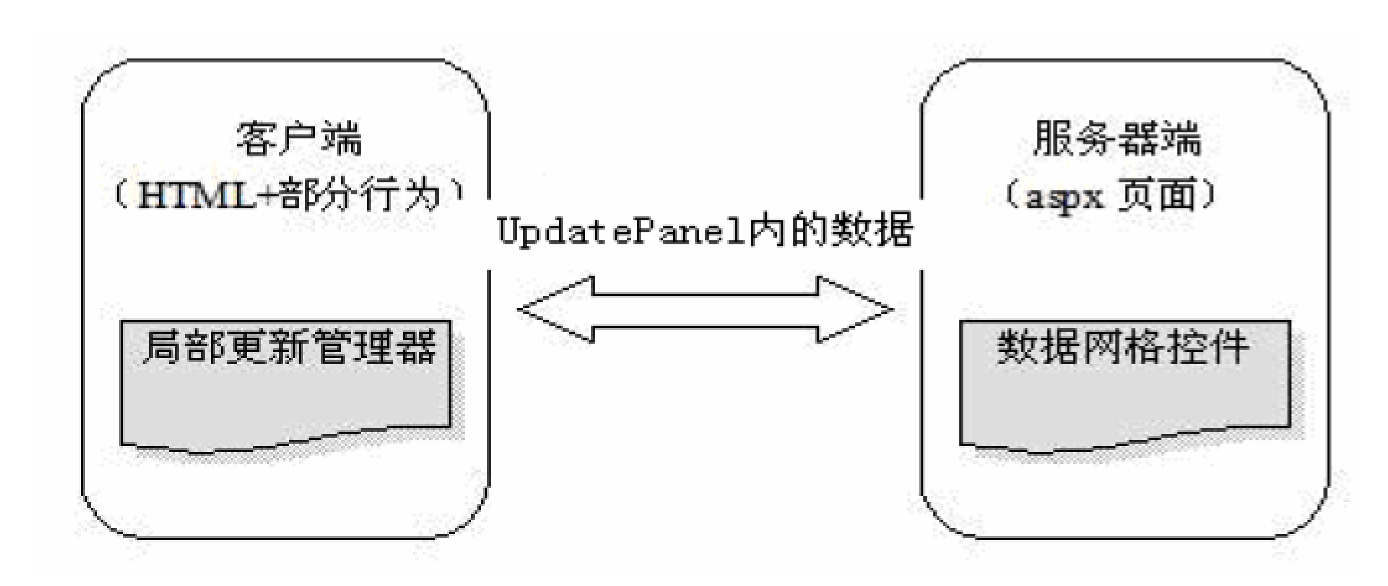


图 24.2 使用 UpdatePanel 控件时，数据交换的情况

从图中可以看出，在使用 UpdatePanel 的 Ajax 网页中，刷新的数据局限于 UpdatePanel 控件的内部，其他部分则与传统的 ASP.NET 编程模型没有区别，依然是先根据客户端的请求计算并取得相应的数据，但与服务器通信时，只将 UpdatePanel 控件内部的内容发送给服务器。

以上是服务器执行 Ajax 任务时最重要的两个控件。

UpdatePanel 控件的内部还可以包括一个子标签<Triggers>。该标签的作用是连接外部的控件来触发内部的刷新。例如，在 UpdatePanel 控件内部有一个 Label1 控件，用来显示当前的时间，外部有一个按钮(Button1)，其 Click 事件的代码如下。

```
 protected void Button1_Click(object sender, EventArgs e)
  {
 Label1.Text = System.DateTime.Now.ToLongTimeString();
}
```

现在要利用外部的按钮来触发内部的刷新，用局部刷新的方式显示时间，可以通过子标签<Trigger>指向外部的 Button 进行触发。其代码如下。

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </ContentTemplate>
    <Triggers>
       <asp:AsyncPostBackTrigger ControlID="Button1" />
     </Triggers>
</asp:UpdatePanel>
```

3. ScriptManagerProxy 控件

ScriptManagerProxy 控件是 ScriptManager 的附加控件。因为每张网页只允许出现一个 ScriptManager 控件，而在有的情况下，例如母版页中已经有了一个 ScriptManager 控件，而在内容页中也需要添加一些额外的对脚本的引用时，就应该在该页面上使用 ScriptManagerProxy 控件。其情况如图 24.3 所示。

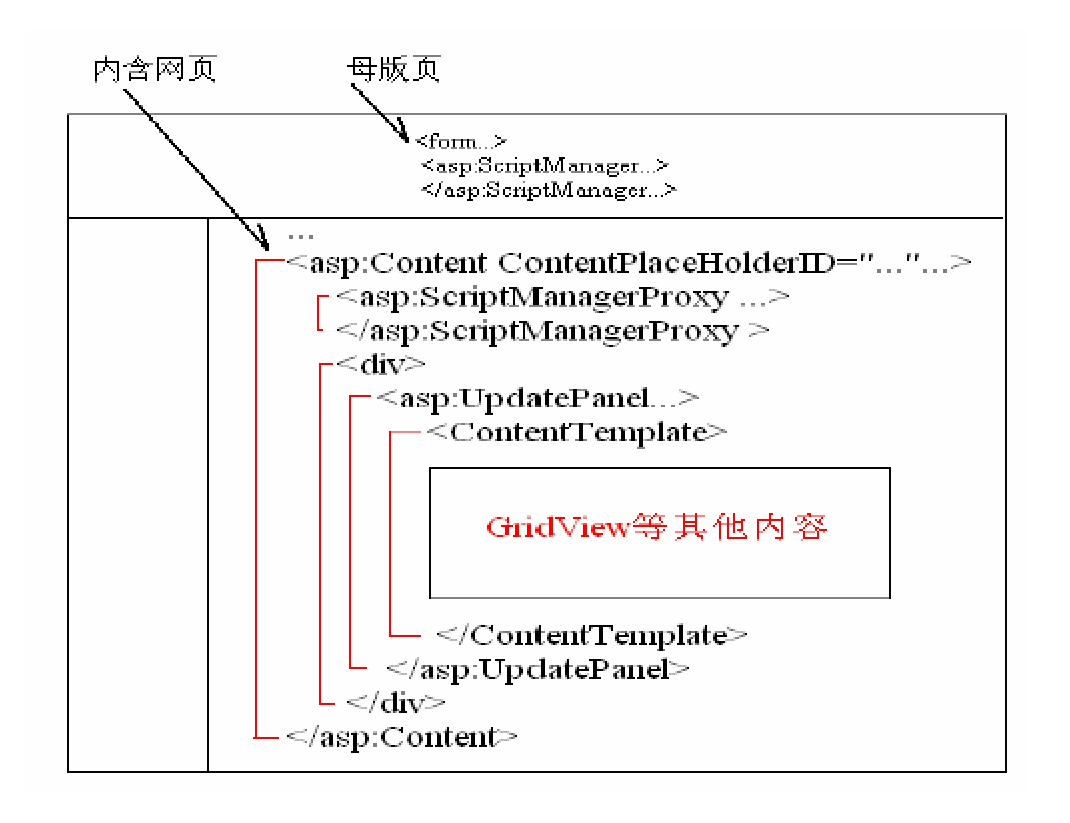


图 24.3　使用 ScriptManagerProxy 控件的情况

图中的母版页中已经使用了 ScriptManager 控件，在嵌入的网页中只能使用 ScriptManagerProxy 控件。

4．UpdateProgress 控件

当网页执行异步通信时，UpdateProgress 控件用来提示异步通信的执行状态。在传统的 Web 应用程序中，浏览器的状态栏将提示当前页面的加载状态。但在局部更新的 Ajax 中，状态栏不再适用了，现在用 UpdateProgress 简单而友好地取代它。

使用 UpdateProgress 控件的代码如下。

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
  <ProgressTemplate>
    <h5>正在执行过程中...</h5>
  </ProgressTemplate>
 </asp:UpdateProgress>
```

如果网页中包括有多个 UpdatePanel 控件，其中任一个 UpdatePanel 被刷新时，都会触发 UpdateProgress 运行，并持续显示本控件中的内容(这里就是显示“正在执行过程中...”)，直到异步通信结束为止。

如果只想用此控件来显示其中某个 UpdatePanel 控件的更新状态时，可以通过该控件的 AssociatedUpdatePanelID 属性与某个 UpdatePanel 控件联系起来。

5. Timer 控件

Timer 控件是一个时间控件，可以为它设置时间间隔(Interval，毫秒为单位)，定时触发某个事件(Tick)，以执行事先设定好的工作。一个很好的用途就是定时刷新 UpdatePanel 控件内的内容。

现在假定设置的条件与前面 Triggers 属性相同，当将 Timer 控件放在 UpdatePanel 控件以内时，每次的 Tick 事件都会触发一次内部刷新。Timer1_Tick 事件代码如下。

```
protected void Timer1_Tick(object sender, EventArgs e)
 {
    Label1.Text = System.DateTime.Now.ToLongTimeString();
 }
```

如果 Timer 控件放在 UpdatePanel 控件以外时，也可以通过 UpdatePanel 控件内部的<Triggers>标签来触发它。

24.4 服务器端 ASP.NET Ajax 应用示例

下面将演示两个使用 ASP.NET Ajax 的 ScriptManager 和 UpdatePanel 控件进行局部刷新的示例。

例 24.1 对比网页整体刷新与局部刷新。

(1) 创建一新网页并转向【设计】视图。

(2) 在工具箱的 Ajax Extensions 选项卡中，双击 ScriptManager 控件，将它增添到网页中来。

(3) 双击 UpdatePanel 控件，将它增添到网页中来。

(4) 先单击 UpdatePanel 控件内部，然后在工具箱中双击 Calendar(日历)控件将它们放置到 UpdatePanel 控件之中。

(5) 单击 UpdatePanel 控件以外的某个网页空间，再将第二个 Calendar(日历)控件放到该处。

(6) 保存并在浏览器中运行该网页。

(7) 改变 UpdatePanel 控件内部的 Calendar 控件的月份时，月份的改变并没有引起整个网页的刷新。

(8) 改变 UpdatePanel 控件外部的 Calendar 控件的月份时，将引起整个网页的刷新。

例 24.2 用 Trigger 触发外面的控件，对 UpdatePanel 内部进行局部刷新。

默认情况下，只能利用内部的控件对 UpdatePanel 进行局部刷新。但也可以通过触发器(Trigger)利用外部控件来刷新 UpdatePanel 内部的数据。具体操作步骤如下。

(1) 创建一张新网页并转向【设计】视图。

(2) 在工具箱的 Ajax Extensions 选项卡中，双击 ScriptManager 及 UpdatePanel 控件，将它们增添到网页中来。

(3) 单击 UpdatePanel 控件内部，然后在工具箱中双击 Label 控件，将它们放置到 UpdatePanel 控件中。

(4) 在 UpdatePanel 控件的外部放入一个 Button 控件。

(5) 双击 Button 控件，创建 Click 事件。

(6) 给 Button 的 Click 事件编写以下代码，以便在 Panel 中显示当前的时间。

```
protected void Button1_Click(object sender, EventArgs e)
   {
     Label1.Text = "当前的时间是: " + DateTime.Now.ToString();
   }
```

(7) 打开【设计】视图，选择 UpdatePanel 控件，然后查看它的属性对话框。

(8) 选择 Triggers 属性，双击右边几个小点，打开 UpdatePanelTrigger 集合编辑器。

(9) 单击【添加】按钮，增加一个新的 Trigger。

(10) 在新 Trigger 的 ControlID 属性的下拉菜单中选择 Button1，然后单击 OK 并保

存网页。

此时在 UpdatePanel 控件中出现了如下代码。

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
    <ContentTemplate>
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </ContentTemplate>
    <Triggers>
       <asp:AsyncPostBackTrigger ControlID="Button1" />
     </Triggers>
</asp:UpdatePanel>
```

(11) 运行网页并多次单击按钮，以查看局部更新的情况。

24.5　给传统网页增添 Ajax 功能

下面通过一个简单的示例讲述给传统的网页添加 Ajax 功能的方法。图 24.4 是一张传统的网页，主要功能是对产品表进行查询和编辑。



图 24.4　一张传统的网页

现在为了添加 Ajax 功能，需执行以下 4 项工作。

(1)　在母版页的<form...>下面增加 ScriptManager 控件。

(2)　在网页的<asp:Content...>下面增加 ScriptManagerProxy 控件。

(3)　将网页的 GridView 控件包含在 UpdatePanel 控件之中，其代码的关系如图 24.5 所示。

(4)　在 UpdatePanel 控件中设 Triggers 标签指向外面的 Button。



图 24.5　代码之间的关系

经过上面的设置就实现了局部刷新和异步传送两大功能，从而改善了客户体验。

24.6 小 结

ASP.NET Ajax 为 Ajax 提供了新控件，简化了设计过程，并能与原来的 ASP.NET 程序无缝地连接，这是它的最大特点。

在新增加的控件中 ScriptManager 与 UpdatePanel 是两个最重要的控件。ScriptManager 用来管理 JavaScript 脚本，每张 Ajax 网页中都需要一个(最多也只能一个)ScriptManager，而且必须将其放在 form 内其他控件之前。UpdatePanel 控件用来确定刷新的范围。两者相结合实现网页的局部刷新功能。其他还有 4 个控件，能够在必要时起重要的辅助作用。

24.7 习 题

1. 选择题

(1) ScriptManager 控件的作用是________。

A. 管理网页　　B. 管理客户

C. 管理脚本　　D. 管理所有的资源

(2) UpdatePanel 控件的作用是________。

A. 定位元素　　B. 确定网页的刷新范围

C. 编辑数据　　D. A+C

(3) UpdateProgress 的作用是________。

A. 编辑数据库　　B. 启动异步通信

C. 连接数据库　　D. 显示异步通信的状态

(4) Timer 控件中的属性 Interval 代表________。

A. 触发事件　　B. 显示时间的格式

C. 时间间隔　　D. 显示时间

2. 判断题

(1) 当不用 ScriptManager 时可以用 ScriptManagerProxy 代替。 ( )

(2) ASP.NET Ajax 与 Ajax 的功能完全相同。 ( )

3. 简答题

(1) 在 UpdatePanel 控件中 Triggers 标签起什么作用？举例说明。

(2) ScriptManager 控件的作用是什么？

(3) ScriptManagerProxy 控件的作用是什么？举例说明。

(4) UpdateProgress 控件的作用是什么？举例说明使用的方法。

4. 操作题

利用 ASP.NET Ajax 控件对传统的 ASP.NET Web 网页增添 Ajax 功能。

第 25 章　Ajax 工具箱

ASP.NET Ajax 工具箱(Control Toolkit)是在微软的倡导下采用“开源”的方式，由各个“社区”(Communities)的开发人员或业余爱好者共同进行开发，并且尽快地进行发布，以便更广泛地发动群众参与，吸取群众智慧。目前已经发布了四十多个比较成熟的成果。微软还将不断发布新成果，这些成果中有的是浏览器端控件，也有的是服务器端控件；有的是对原有控件功能的扩展或增强，也有的是独立的新控件。

25.1　安装 ASP.NET Ajax 工具箱

为了使用 ASP.NET Ajax 工具箱中的控件，需要进行下载和安装。其步骤如下。

(1)　下载并解压 Toolkit 控件。

在因特网中找到 AjaxControlToolkit Binary .NET 35.zip 文件进行下载，并将其解压到指定的目录中。

注意： 下载时参考的 URL 是 http://ajaxcontroltoolkit.codeplex.com/releases/view/33804。特别注意下载的版本为.NET 35，其他版本在本系统中可能不能运行。

(2)　安装 Toolkit 控件。

安装步骤如下。

①　任意创建一个网站，并打开一张网页(说明：这里安装的 Toolkit 能在各个网站和网页中共享)。

②　在网站工具箱中增添一页以便放置 Ajax Toolkit 工具。方法是，右击工具箱，在弹出的菜单中选择【添加选项卡】命令，然后给选项卡取名。例如取名为 Toolkits。

③　右击选项卡名，选择【选择项】命令，打开【选择工具箱项】对话框，如图 25.1 所示。

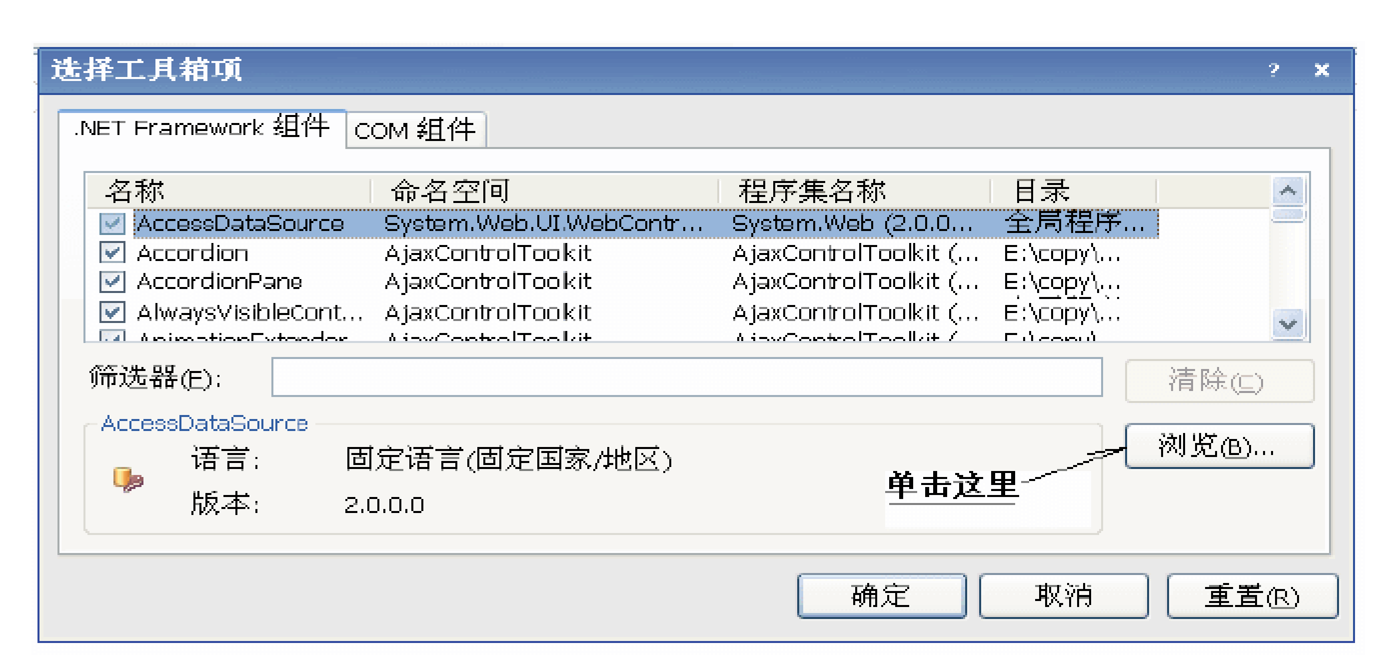


图 25.1　【选择工具箱项】对话框

④　单击【浏览】按钮，然后找到前面下载后解压的目录，双击 AjaxControlToolkit.dll

文件，单击【确定】按钮。

经过前面的操作，网页的工具箱的选项卡下面将自动增加 40 多个 Toolkit 控件。这些控件就是 Ajax 工具箱控件。这些控件的作用及使用方法都可以从因特网中查到，参考的网址是 http://ajax.asp.net/ajaxtoolkit/。

下面介绍几个利用工具箱设计控件的方法。

25.2 设计 Accordion(可折叠面板)控件

Accordion 是手风琴的意思。Accordion 控件的特点是能像手风琴那样折叠或展开。当网页中某些列表过长不便于摆放和阅读时，可以先将它们归类到不同的面板中，然后放到这个控件中来。其实网站的工具箱本身使用的就是这种结构。

25.2.1 Accordion 的嵌套结构

设计 Accordion 控件的关键是，将文档正确地归类并组织成嵌套的结构。

假定将三个面板折叠或展开，其嵌套的代码如下。

```
<form id="form1" runat="server">
<div>
<asp:ScriptManager ID="ScriptManager1" runat="server" />
<asp:Accordion ID="Accordion1" runat="server">
 <Panes>
<!—第一块面板开始-→
 < asp:AccordionPane ID="AccordionPane1" runat="server">
    <Header>
       <a href="" onclick="return false">第一页</a>
    </Header>
    <Content>
    <p> 这是第一块面板</p>
    </Content>
  </asp:AccordionPane>
<!--第二块面板开始→
   <asp:AccordionPane ID="AccordionPane2" runat="server">
   <Header>
  <a href="" onclick="return false">第二页</a>
    </Header>
  <Content>
   <p> 这是第二块面板</p>
  </Content>
 </asp:AccordionPane>
<!--第三块面板开始→
  <asp:AccordionPane ID="AccordionPane3"
runat="server">
 <Header>
   <a href="" onclick="return false">第三页</a>
  </Header>
    <Content>
        <p> 这是第三块面板</p>
    </Content>
   </asp:AccordionPane>
```



图 25.2 简单的可折叠界面

```
  </Panes>
 </asp:Accordion>
 </div>
</form>
```

结果显示如图 25.2 所示。

代码中需要注意以下 4 个问题。

- 每个应用 Ajax 的网页都要首先放入一个 ScriptManager 控件，其作用主要是当网页打开时将相关的 JavaScript 代码从服务器送到浏览器，否则控件无法运行。
- 各面板的代码都被嵌入到 Accordion 控件中，每一块面板用 AccordionPane 控件表示。控件中包括两个字段。

```
<Header>
</Header>
```

与

```
<Content>
</Content>
```

前者用于编写标题，后者编写实际内容。

- 各个面板的 Header 中用以下代码来实现折叠和展开的操作。

```
<Header>
    <a href="" onclick="return false">第＊页</a>
</Header>
```

- Accordion 控件继承于 System.Web.UI.WebControls.WebControl，因此它一产生就拥有了该控件的所有属性/方法/事件。声明 Accordion 控件时所常用的属性标签如表 25.1 所示，这些属性将主要写在 Accordion 控件的标签中。

表 25.1　Accordion 控件的常用属性

属性标签名	描　　述
SelectedIndex	该控件初次加载时展开的 AccordionPane 面板的索引值
HeaderCssClass	该 Accordion 中包含的所有 AccordionPane 面板的标题区域所应用的 CSS Class
ContentCssClass	该 Accordion 中包含的所有 AccordionPane 面板的内容区域所应用的 CSS Class
AutoSize	在展开具有不同高度的 AccordionPane 面板时，该 Accordion 的总高度的变化方式。可选如下 3 个值。 • r None：该 Accordion 将随着当前展开的 AccordionPane 面板的高度自由伸长/缩短。 • r Limit：该 Accordion 将随着当前展开的 AccordionPane 面板的高度自由伸长/缩短，不过最高不会超过 Accordion 的 Height 属性设定值。若是其内容高度超过了 Height 属性设定值，将自动显示滚动条。 • r Fill：该 Accordion 的高度将固定为 Height 属性的设定值，不随当前展开的不同高度的 AccordionPane 面板而变化。若是某个 AccordionPane 的内容高度超过了 Height 属性设定值，则将自动显示滚动条
FadeTransitions	若该属性值设置为 true，则在切换当前展开的 AccordionPane 面板时，将带有淡入淡出效果

续表

属性标签名	描　　述
TransitionDuration	展开/折叠一个 AccordionPane 面板的过程所花费的时间，单位为毫秒
FramesPerSecond	播放展开/折叠 AccordionPane 面板动画的每秒钟帧数
DataSourceID	页面中某个 DataSource 控件的 ID，用于通过数据绑定自动生成 AccordionPane 面板
<Panes>	该标签内将包含一系列的<ajaxToolkit:AccordionPane>标签，即 Accordion-Pane 的声明，用来表示 Accordion 中包含的面板
<HeaderTemplate>	在使用数据绑定功能自动生成 AccordionPane 面板时，该标签内将定义每个面板的标题区域的内容模板
<ContentTemplate>	在使用数据绑定功能自动生成 AccordionPane 面板时，该标签内将定义每个面板的正文区域的内容模板

25.2.2　Accordion 控件的应用示例

Accordion 控件的应用示例如图 25.3 所示。



图 25.3　Accordion 控件的应用示例

上述代码的显示结果如图 25.4 所示。

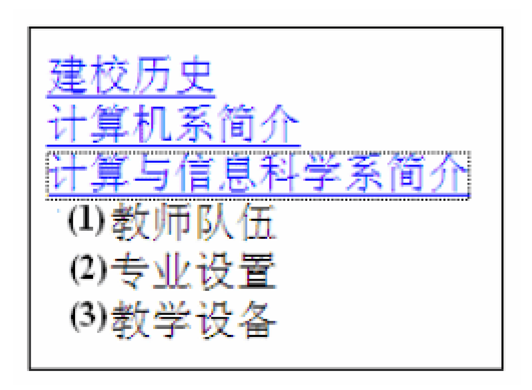


图 25.4　运行结果

只要单击标题该模板即可展开，其他模板则自动折叠。

下面再介绍几个支持原有控件的新控件。这些控件都是用 JavaScript 语言开发的，它们之中有的是为了改善原有控件的外貌，有的是给原有控件增添新功能。利用 Toolkit 控件来设计它们将变得更加直观和容易。

25.3　几个支持 Button 的 Toolkit

为了使用 Toolkit 控件，首先要在网页的 form 标签下，所有其他控件的前面增添 ScriptManager 控件。

增添一个 Button 控件，然后右击该控件，选择【添加扩展程序】命令，将打开另一个对话框并显示出一连串的图标，表示多个支持 Button 控件的 Toolkit 控件，如图 25.5 所示。

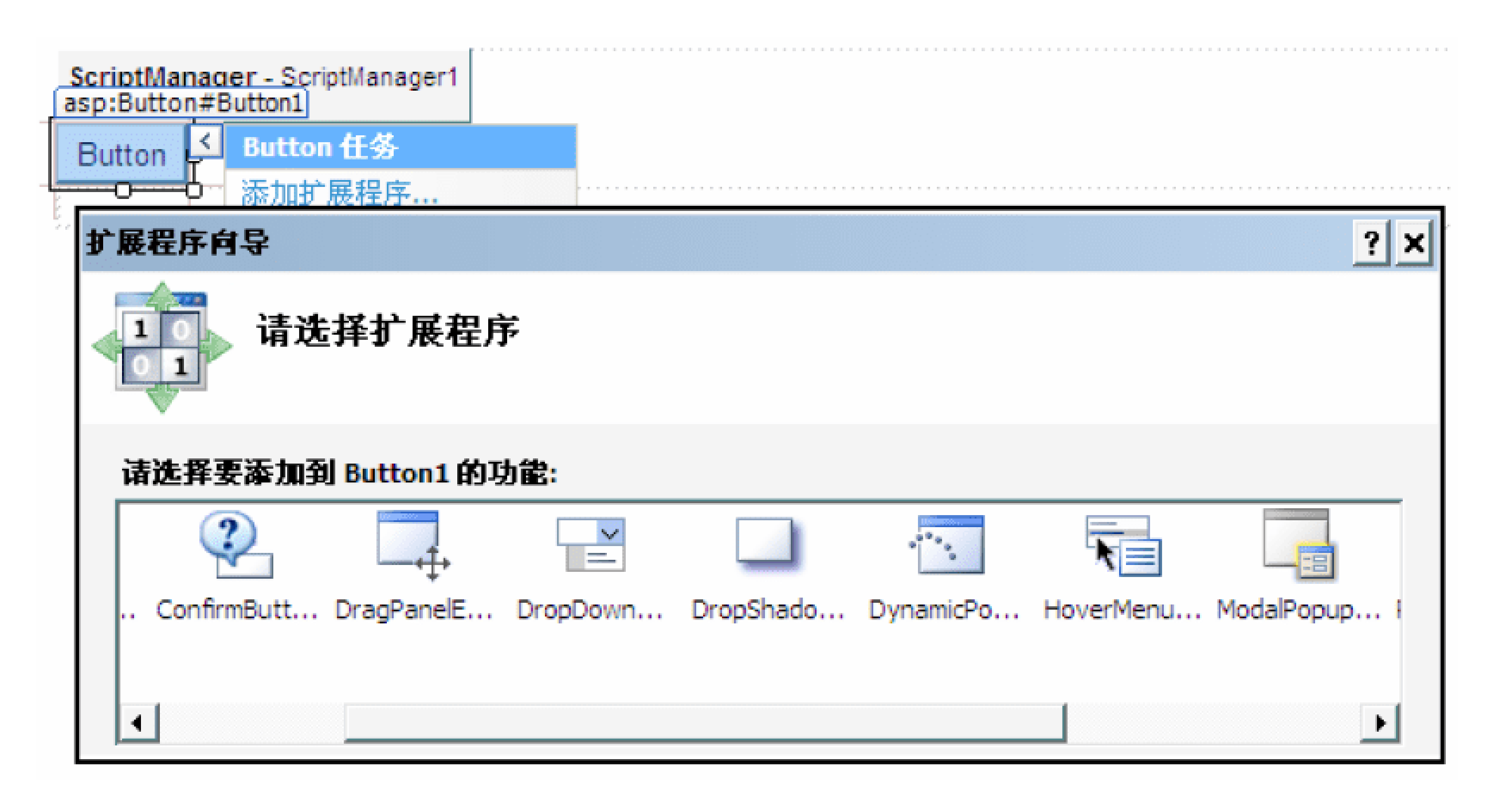


图 25.5　支持 Button 控件的几个 Toolkit 控件

25.3.1　用于增加【确认】功能的 Toolkit

为了给按钮增加执行前最后确认的功能，双击ConfirmButtonExtenter控件的图标，将自动生成一个Toolkit控件。将该控件的TargetControlID属性指向被支持的控件(这里就是指向Button1)。然后可以根据需要在ConfirmText的属性中填写提示信息。这里填写的是“确定

了吗？”。此时控件将自动生成以下代码。

```
<asp:Button ID="Button1" runat="server" Text="Button" />
<cc1:ConfirmButtonExtender ID="Button1_ConfirmButtonExtender"
runat="server"
ConfirmText="确定了吗？" Enabled="True" TargetControlID="Button1">
</cc1:ConfirmButtonExtender>
```

程序运行中当单击 Button 按钮时，将弹出确认提示，如图 25.6 所示。



图 25.6　实现确认功能

25.3.2　为控件增强立体感

在控件的背景中适当地增加阴影可以增强立体感。为此可以使用 DropShadowExtender 控件。该控件有几个重要的属性。

- TargetControlID：支持的对象。
- Width：阴影的宽度，默认为 5 像素。
- Opacity：阴影的透明度，从 0(完全透明)～1(完全不透明)。
- trackPosition：阴影是否随控件位置的变动而改变。

代码如下。

```
<asp:Button ID="Button1" runat="server" Text="Button" />
    <cc1:DropShadowExtender ID="Button1_DropShadowExtender"
runat="server"
          Enabled="True" TargetControlID="Button1" Width="3" Radius="4">
    </cc1:DropShadowExtender>
```

显示结果如图 25.7 所示。

Button

图 25.7　增加阴影

25.4　使用几个支持 TextBox 的控件

25.4.1　对输入的数据类型进行过滤

为了限制输入的类型，可以选用 FilteredTextBoxExtender 控件。该控件有一个关键属性：FilterType，它代表允许输入的数据类型，共包括 4 种类型，它们是：小写字符(LowerCaseLetters)、大写字符(UpperCaseLetters)、数字(Numbers)和自定义(Custom)。

25.4.2　用按钮方式增减输入的数字

双击 NumericUpDownExtender 图标后，给控件填写(或修改)属性。几个关键的属性如下。

- TargetControlID：指向被支持的控件。
- Maximum：最大值。
- Minimum：最小值。
- RefValues：一组由数字或字符串组成的枚举型变量。
- Width：被支持控件的宽度，默认值为 0。
- Step：步进，默认为 1。

控件的外貌如图 25.8 所示。

图 25.8　增减数字控件

如果想用 TextBox 控件来输入年龄，年龄限制在 18～31 岁之间，用一个 NumericUpDownExtender 控件来约束输入。其代码的设置如下。

```
<cc1:NumericUpDownExtender ID="TextBox1_NumericUpDownExtender"
runat="server" Enabled="True" Width="50"
RefValues="18;19;20;21;22;23;24;25;26;27;28;29;30;31"
    TargetButtonDownID="" TargetButtonUpID=""
TargetControlID="TextBox1">
    </cc1:NumericUpDownExtender>
```

如果想用 NumericUpDownExtender 来限制 TextBox 只能输入星期几时，设置的代码如下。

```
<cc1:NumericUpDownExtender ID="TextBox1_NumericUpDownExtender"
runat="server"  Enabled="True" Width="150"
RefValues=";星期日;星期一;星期二;星期三;星期四;星期五;星期六;"
TargetButtonDownID="" TargetButtonUpID=""
     TargetControlID="TextBox1" >
  </cc1:NumericUpDownExtender>
```

25.5　小　　结

ASP.NET Toolkit 工具箱是在微软的组织下，由社区的设计者或者业余爱好者共同开发的。这样做既能集中群众的智慧，又能引起大家的兴趣。目前已经发布了 40 多种 Toolkit 控件，供设计者使用。这些控件有的是新控件，有的是进一步美化原有控件或者给原有控件增添新功能。这些控件如果让设计者自行设计比较困难，而且很费时间和精力，直接调用 Toolkit 控件将大大简化这一过程。

调用 Toolkit 控件的关键是下载并安装适合自己系统的版本，并按控件的要求设置参数。具体方法在因特网上都可以查到。

25.6 习　　题

1. 判断题

(1) Ajax Toolkit 只是一些服务器端控件。 (　　)

(2) Ajax Toolkit 只是一些浏览器端控件。 (　　)

(3) Ajax Toolkit 既可能是一个新控件，也可能是对原有控件功能的增强。 (　　)

2. 操作题

(1) 利用 Accordion 控件设计一张网页。

(2) 模仿前面的示例，用 3 种 Toolkit 控件给 Button、TextBox 控件增强功能。

(3) 查阅因特网找到 3 种 Toolkit 的使用方法。

第六部分

Web 服务

在前面讲述动态网站的开发技术中，都是将网站看成一个相对独立的实体。虽然这个实体面临来自各类客户的不同要求，但都要依靠本网站的资源来解决问题。这种“孤岛”现象能否改进，网站的开发模式能不能有新的突破？

实际上，一个网站不需要，有时也不可能完全依靠自己的力量来解决问题。站在更高的角度来思考，将整个因特网看成一个开发平台，采取资源共享的方式。这就是说，一个网站可以借助于第三方(或多方)的力量来共同解决问题，这样会大大减少重复劳动，降低开发成本，并提高开发效率。

Web 服务的思想由此产生，它是 Web 的一场革命。在这种思想的引导下，一批先进的理念，如“软件即服务”、“网络即计算机”等诞生了，一批面向服务架构的网站(Service Oriented Architecture，SOA)出现了。经过几年的实践，XML Web 信息服务在这方面取得较大的成功，其他方面也有很多进展，但也有些方面并没有达到预期的效果。

受篇幅限制，本书中将集中介绍 XML Web 服务技术。

第 26 章　XML Web 服务

本章先讲述 XML Web 服务的一般概念，然后通过几个典型示例(气象预报、航班信息、股票信息、电视台节目预报)重点讲述调用 XML Web 服务的方法，最后再讲述创建 XML Web 服务网站的方法。具体内容包括:

- XML Web 服务的特点。
- XML Web 服务的过程。
- 相关协议。
- 几个典型的应用。
- 创建 XML Web 服务网站。

26.1　XML Web 服务的特点

在人们的生活中，依靠服务的现象非常普遍，比如到餐馆去就餐，你只要按照餐馆提供的菜谱点菜，饭后付款就可以了，其他一切都不必详细了解。

Web 服务也是这样，作为服务者(Web Service)，将公开一些服务项目，主要包括提供信息服务或承担某种计算任务，通常不提供显示界面方面的服务。

作为消费者(Web Consumers)，他既可能是 Web 的页面，也可能是 Windows 窗体，还可能本身又是另一方的服务者。他只要按照服务方提供的调用命令和参数进行调用，并将调用结果(数据)插入到自己的应用中即可。这个调用的过程代表着一种新的计算模式，有时被称为“网络计算”。

Web 服务只有能够在整个因特网的范围内发挥作用时，才能充分发挥服务的效益。这就是说，Web 服务不仅要能跨越机器的界限，还要能不受操作系统、开发平台、开发语言的限制。不论服务者和消费者各采用什么样的操作系统(Windows、UNIX、Linux 等)、什么样的开发平台(.NET、J2EE、Rail 等)，使用什么样的程序语言(C++、C#、Java、Delphi、Ruby)等，都能够在彼此之间开展服务工作。

为了实现上述目标，Web 服务只能建立在高度抽象的“松耦合”的基础之上。构成服务与消费关系的必要条件不是机器、语言、系统和平台，而是一些因特网中通用的通信标准，如 XML、SOAP、HTTP 等，和服务双方必须遵守的“契约”(Contracts)。

26.2　XML Web 服务的过程

XML Web 服务是一个比较复杂的过程。其过程大体是:

(1)　消费者找准服务者及服务项目，然后按照要求调用服务。

(2)　服务者接受申请并完成服务后，将信息序列化后发出。

(3)　消费者接受传来的信息，还原序列化的信息后再插入到消费者的应用之中。

为了简化上述过程，系统在消费端增加了一个中间环节——“代理”(Proxy)。“代理”是服务端驻留在消费端的代表。实际上它是一组类，并在消费端进行了注册(每种服务需要注册一个代理)，用来自动完成调用服务时一系列的复杂工作。当消费端需要调用服务时只需要直接找代理，由代理再去连接服务端，并发出请求信息，然后对返回的信息自动完成还原序列化的工作。

这样，从消费端调用服务的过程，就如同对自己内存中某些对象的调用一样，变得十分简单。调用的过程如图 26.1 所示。

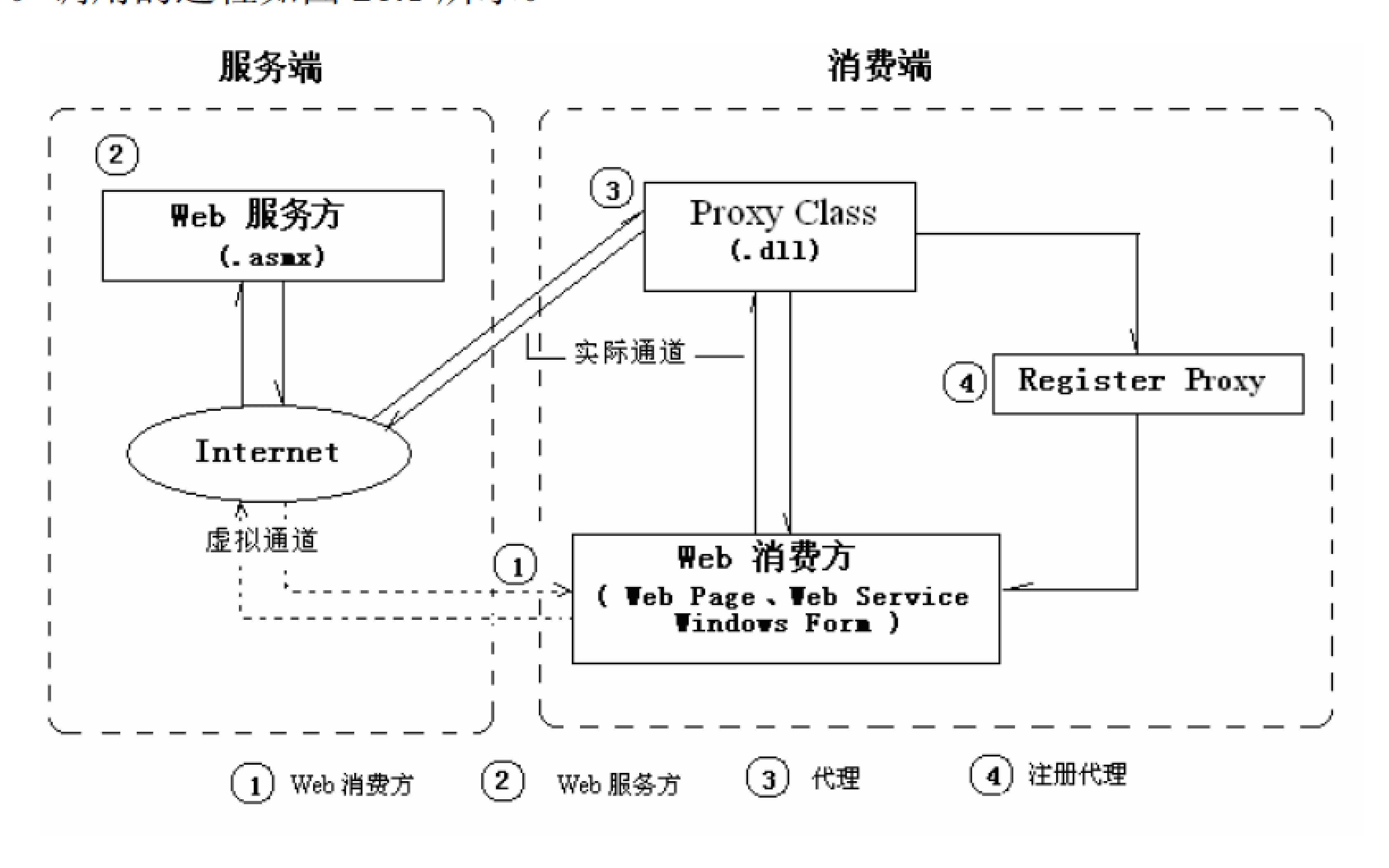


图 26.1　通过代理调用服务的过程

图 26.1 表明，在人们的感觉中，调用服务的过程，似乎是 Web 消费方在与 Web 服务方直接进行通信(图中的虚拟通道)。而实际上却是通过代理(Proxy Class)进行通信的(图中的实际通道)。代理是服务方驻留在消费方的代表，并且已经在消费端进行了注册(见图中的 Register Proxy)。

26.3　相 关 协 议

Web 服务的过程需要用到一些因特网的通用标准，主要包括以下几项。

26.3.1　发现服务

为了调用服务，首先要有一个发现服务的过程。即客户应用程序如何找到并检查需要的服务。服务者也应该告诉客户接受服务时的要求(有时这是一种有偿服务)。

UDDI(the Universal Description, Discovery and Integration)是一套基于 Web 分布式的，为 Web Services 提供的信息注册的标准规范。服务的提供者可以通过它宣传自己提供的服务项目以及服务的要求；服务的消费者可以通过它找到需要的服务者，了解服务的要求。其网址为 http://www.uddi.org/。目前已有数百家公司注册。

26.3.2 传输信息

SOAP(Simple Object Access Protocol，简单对象访问协议)是交换 XML(或 JSON)编码信息的轻量级协议。它包括三个主要方面：XML-Envelope，它像一个信封，描述信息内容以及如何处理内容的框架；将程序对象编码成 XML(或 JSON)对象规则；执行远程调用(PRC)的约定等。SOAP 可以运行在任何其他传输协议之上，通常运行在 HTTP 之上。纯文本的 XML(或 JSON)用来传输大多数数据类型是非常理想的，但如果除传输 XML(或 JSON)信息以外，还需要传输命令或指令时，使用 SOAP 更为适用。

26.3.3 解释信息

WSDL(Web Services Description Language)是一种基于 XML 格式的文档，用的是一种计算机和人都易于阅读的方式来描述 Web Services 中的函数，包括方法名、需要提供的参数以及返回的数据类型等。调用服务的设计者需要认真阅读这些文件，以便编写出正确的调用程序。

26.3.4 WS-*规范

SOAP 与 WSDL 的主要特性之一在于它们都是可扩展的。微软、IBM、BEA 几个市场巨头联合在一起，组成了一个 WS-I (Web Service Interoperability)的组织。由于他们的一系列规范的名称都以 WS-开头，并以一个表明设计目的的表达式作为结束，所以这些规范被命名为 WS-*(读成 WS Star)。该规范涵盖安全、描述、发现、消息传递和协调与事务等多个方面。

26.4 几个典型的应用

网站中常常需要用到一些动态信息，有些信息不仅量大，而且经常变化。若采用常规的办法进行管理，不仅需要给数据库不断输入和更新大量信息，还要有众多的专业人员进行维护。即使这样，也难以保证信息的及时性和准确性。现在使用 XML Web Service，上述难题都能够迎刃而解。人们普遍认为这是 XML Web 服务中应用得最为成功的领域。

下面将通过几个典型的应用，来说明调用 XML Web Service 的方法。为了具体地说明这些方法，我们将借用国内一个很好的公用事业网站 WebXml 进行讲解。该网站的网址是 http://www.webxml.com.cn/zh_cn/index.aspx。

截至 2010 年 12 月底，该网站已经提供了很多免费的服务项目，包括气象预报、航班或列车时刻表、货币转换、电视台节目预告以及各类股票信息等 20 多项服务。其首页的界面如图 26.2 所示。

网站中的服务信息有的来自其他服务网站，如中国气象局网站、外汇管理局网站以及多种股票的网站等。WebXml 网站本身既是服务者，又是前面那些网站的消费者。



图 26.2　WebXml 网站的首页界面

26.4.1　调用气象预报服务

1. 显示界面的设计

气象预报显示的界面如图 26.3 所示。

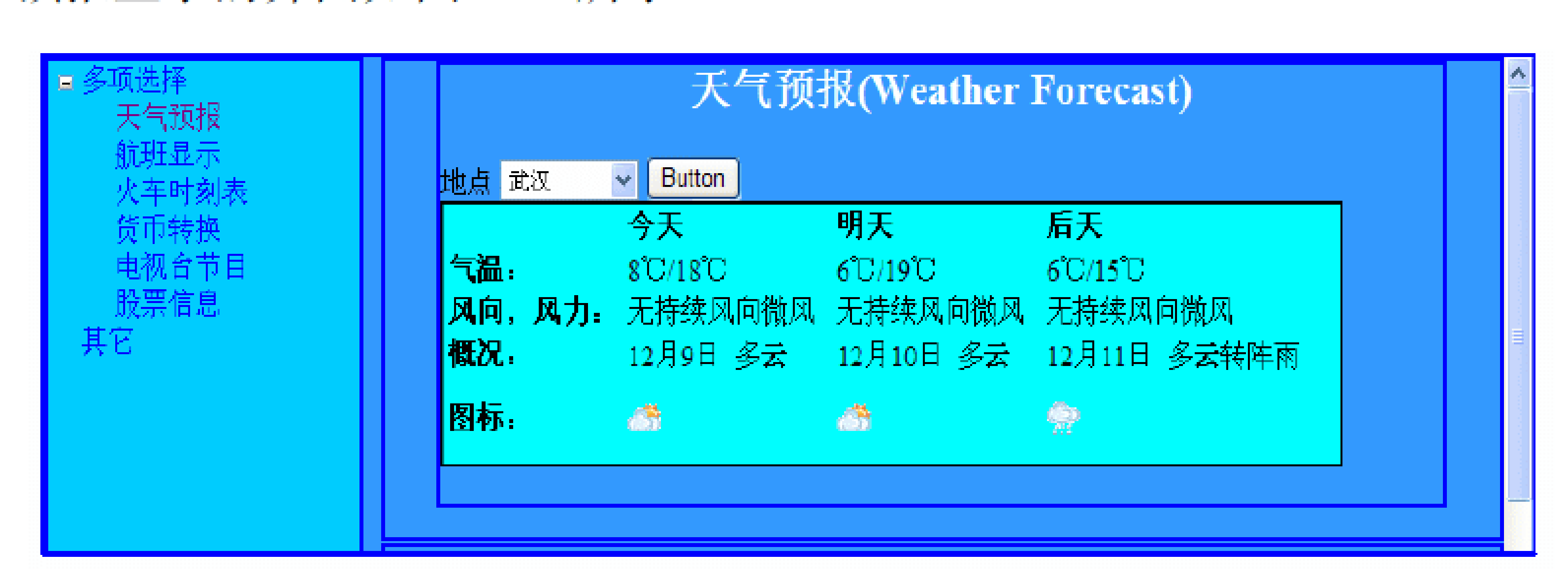


图 26.3　气象预报显示界面示例

气象预报服务网站能够提供 2500 个城市(其中国内 2400 个，国外 100 个)5 天的气象预报信息。每 2.5 小时更新一次。只要输入城市名(如北京、长沙、New York 等)，就可以输出该城市的气象预报。

2. 设计步骤

设计的步骤如下。

(1) 创建一个 ASP.NET 网站。

(2) 找到提供服务的网站，打开服务项目后，查看其中的“接口帮助”文档。在该文档中提供了.asmx 文档的 URL、多种调用方法、每种方法需要提供的参数以及返回的数据类型等。该文件很大，现将有关片段摘录到图 26.4 及图 26.5 中。

天气预报 WEB 服务 接口说明
www.webxml.com.cn

Endpoint: http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx
Disco: http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx?disco
WSDL: http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx?wsdl

简要说明:

- 为让更多的开发人员学习WEB服务开发，此服务支持免费用户使用（说明：免费用户24小时内查询不超过50次并且获取二次数据大于间隔 600ms），注册会员可获得试用。
- 为支持多种平台开发，此WEB服务接口提供了多种返回类型可选择，包括.NET的…

图 26.4　天气预报接口说明文档片段(1)

```
返回数据：一维字符串数组 String()，结构如下：
Array(0) = "省份 地区/洲 国家名（国外）"
Array(1) = "查询的天气预报地区名称"
Array(2) = "查询的天气预报地区ID"
Array(3) = "最后更新时间 格式：yyyy-MM-dd HH:mm:ss"
Array(4) = "当前天气实况：气温、风向/风力、湿度"
Array(5) = "第一天 空气质量、紫外线强度"
Array(6) = "第一天 天气和生活指数"
Array(7) = "第一天 概况 格式：M月d日 天气概况"
Array(8) = "第一天 气温"
Array(9) = "第一天 风力/风向"
Array(10) = "第一天 天气图标 1"
Array(11) = "第一天 天气图标 2"
Array(12) = "第二天 概况 格式：M月d日 天气概况"
Array(13) = "第二天 气温"
Array(14) = "第二天 风力/风向"
Array(15) = "第二天 天气图标 1"
Array(16) = "第二天 天气图标 2"
 …
```

图 26.5　天气预报接口说明片段(2)

在图 26.4 所示的片段中提供了 WeatherWS.asmx(这是一个提供服务的类文件)的网址(见图 26.4 中 Endpoint 行)。

图 26.5 说明，如果选择 getWeather()方法时，将返回“一维字符串数组”。各种数据分配在数组的不同字段中。

(3) 取出接口说明文件中.asmx 文件(WeatherWS.asmx)的网址(即图 26.4 中被选择的网址)。

(4) 返回到网站中，创建并注册服务代理 (Proxy)。其方法如下。

① 右击网站名，在弹出的快捷菜单中选择【添加 Web 引用】命令，打开的对话框如图 26.6 所示。



图 26.6　【添加 Web 引用】对话框

② 按照图中显示的顺序创建服务代理并进行注册。

a. 将.asmx 文件的网址(URL)复制到这里。

b. 单击【前往】按钮。

c. 给【Web 引用名】改用有意义且容易记忆的名字。

d. 单击【添加引用】按钮。

经过编译后，再打开 Web.config 文件时，就可以看见 Proxy 的注册代码。

```
<appSettings>
<add key="getweather.WeatherWS"
value="http://webservice.webxml.com.cn/WebServices/WeatherWS.asmx"/>
</appSettings>
```

后面几步工作都具体体现在下面的网页代码中。

(5) 在网页的代码文件中，生成代理对象。

(6) 通过代理对象调用服务中的方法。

(7) 由于该服务返回的类型是“一维字符串数组”，因此可以用 string[]接收返回的结果。

(8) 为了显示气象图标，需要先将图标下载、解压后放到网站中来。下载的网址是 http://www.webxml.com.cn/images/weather.zip。

(9) 在网页中拖入若干 Label 控件和 Image 控件(参照显示界面的配置)。代码如下。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
public partial class Weather : System.Web.UI.Page
{
    weather.WeatherWS proxy1;                //其中 weather 是 Web 引用名
    String[] weathers;
    string cityCode;
    protected void Button1_Click1(object sender,  EventArgs e)
    {
    //5
        proxy1 = new weather.WeatherWS();    //生成代理对象
```

```
        //将下拉菜单中选择的城市放入字符串数组中
        cityCode = DropDownList1.SelectedItem.ToString();
    //6
        weathers = proxy1.getWeather(cityCode, "");  //调用方法 getWeather()
          //7
        Label1.Text = weathers[8];           //第一天的气温
        Label2.Text = weathers[13];          //第二天的气温
        Label3.Text = weathers[18];          //第三天的气温
        Label4.Text = weathers[9];           //第一天的风力/风向
        Label5.Text = weathers[14];          //第二天的风力/风向
        Label6.Text = weathers[19];          //第三天的风力/风向
        Label7.Text = weathers[7];           //第一天概况
        Label8.Text = weathers[12];          //第二天概况
        Label9.Text = weathers[17];           //第三天概况
          //8, 9
        Image1.ImageUrl = weathers[10];       // 第一天的气象图标
        Image2.ImageUrl  =  weathers[15];     // 第二天的气象图标
        Image3.ImageUrl  =  weathers[20];     // 第三天的气象图标
    }
}
```

注意，数组中的下标值是根据“接口说明”文档中的提示(见图 26.5)来确定的。

26.4.2 调用国内航班信息

在 Web 服务网站中选择【国内飞机航班时刻表 WEB 服务】项，获得的界面如图 26.7 所示。



图 26.7 国内航班 Web 服务

从.asmx 文件中得知其服务的类名是 DomesticAirline.asmx，从接口帮助文件中得知调用的方法名是 getDomesticAirlinesTime。要求输入的参数和返回的数据类型如下。

- 输入参数：startCity = 出发城市(中文城市名称或缩写，空则默认为上海)；lastCity = 抵达城市(中文城市名称或缩写，空则默认为北京)；theDate = 出发日期(String 格式：yyyy-MM-dd，如 2007-07-02，空则默认为当天)；userID = 商业客户 ID(免费客户不需要)。
- 返回数据：DataSet，Table(0)结构为 Item(Company)航空公司、Item(AirlineCode)航班号、Item(StartDrome)出发机场、Item(ArriveDrome)到达机场、Item(StartTime)出发时间、Item(ArriveTime)到达时间、Item(Mode)机型、Item(AirlineStop)经停、Item(Week)飞行周期(星期)。

根据上述要求可以用两个 TextBox 控件来输入起飞城市(不填写时代表上海)和到达城市(不填写时代表北京)；用一个 TextBox 控件来输入日期，格式为 yyyy-mm-dd，如 2007-

07-03 (不填时代表当天)。由于返回的类型是数据集(DataSet,Table(0))，因此可以将其作为数据源，用 GridView 控件来显示结果。

给代理(Proxy)注册的过程与其他设计方法与调用气象预报服务相同。编写的代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
   airline.DomesticAirline air = new airline.DomesticAirline();
   DataSet d = air.getDomesticAirlinesTime(TextBox1.Text, TextBox2.Text,
TextBox3.Text,"");
   DataTable ta=d.Tables[0];

   GridView1.DataSource =ta;
   DataBind();
}
```

显示的航班信息界面如图 26.8 所示。

多项选择
天气预报
航班显示
电视节目预告

起始城市 长沙　终点城市 成都　时间　Button

Company	AirlineCode	StartDrome	ArriveDrome	StartTime	ArriveTime	Mode	AirlineStop
厦门航空有限公司	MF8401	黄花机场	双流国际机场	09:10	10:55	737	0
中国南方航空(集团)公司	CZ3461	黄花机场	双流国际机场	14:05	15:50	320	0
四川航空股份有限公司	3U8998	黄花机场	双流国际机场	15:05	16:50	JET	0
厦门航空有限公司	MF8471	黄花机场	双流国际机场	18:05	19:40	737	0
成都航空公司	EU2206	黄花机场	双流国际机场	21:10	22:55	320	0

图 26.8　国内航班信息

26.4.3　调用股票信息

WebXml 服务网站提供了几种股票信息的服务。假定选择了【中国股票行情数据 WEB 服务】选项。单击.asmx 文件，打开 ChinaStockWebService 对话框。其中提供了几种不同的调用方法，有的返回数据，有的返回图形。

1. 显示股票的数字信息

为了显示股票的数字信息，选用 getStockInfobyCode()方法，该方法的说明如图 26.9 所示。

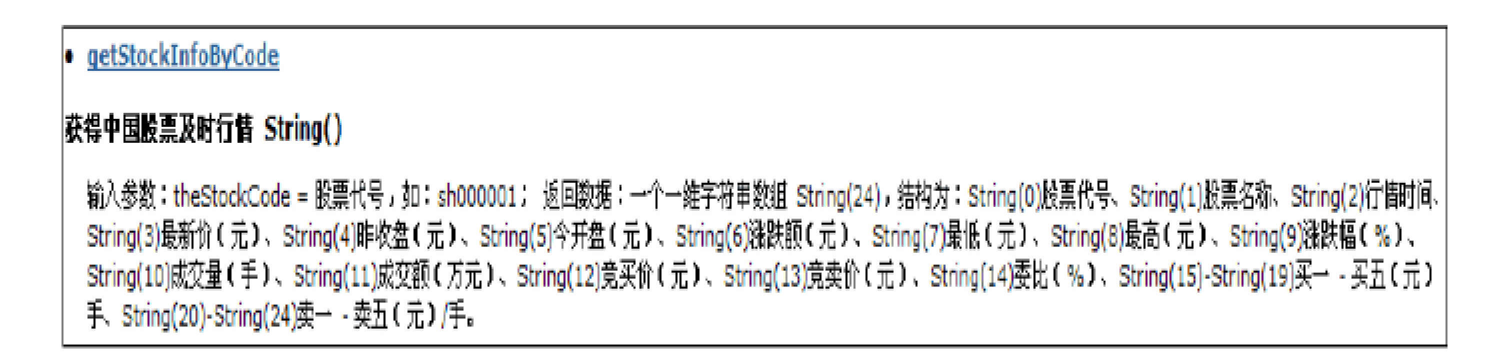


图 26.9　显示股票数字信息的方法说明

它要求输入股票代号，返回一个字符串数组。设计过程与“气象预报”基本相同。代

码如下。

```
    protected void Button1_Click(object sender, EventArgs e)
      {
          stockimage.ChinaStockWebService proxy1 = new
stockimage.ChinaStockWebService();
          string stockCode = TextBox1.Text;
          string[] pp = proxy1.getStockInfoByCode(stockCode);
          Label1.Text = pp[0];
          Label2.Text = pp[1];
          Label3.Text = pp[2];
          Label4.Text = pp[3];
          Label5.Text = pp[4];
          Label6.Text = pp[5];
          Label7.Text = pp[6];
          Label8.Text = pp[7];
          Label9.Text = pp[8];
          Label10.Text = pp[9];
          Label11.Text = pp[10];
          Label12.Text = pp[11];
          Label13.Text = pp[12];
          Label14.Text = pp[13];
          Label15.Text = pp[14];
        }
```

股票信息的数字显示结果如图 26.10 所示。

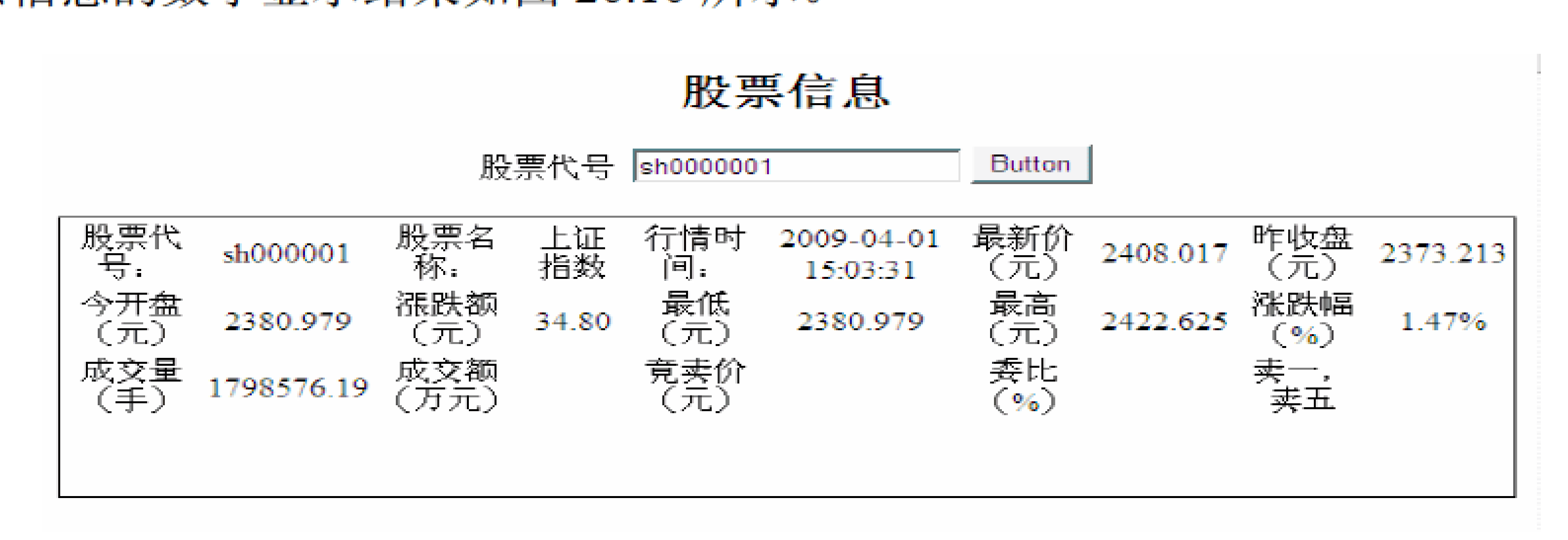


图 26.10　股票信息的数字显示

2. 股票的图形显示

为了用图形显示股票信息，选用 getStockImage_KByteByCode()方法，该方法的要求如图 26.11 所示。

- getStockImage_kByteByCode

获得中国股票GIF日/周/月 K线图字节数组

输入参数：theStockCode = 股票代号，如：sh000001；theType = K 线图类型（D：日[默认]、W：周、M：月），返回数据：股票GIF K 线图字节数组。

图 26.11　用图形显示股票信息的要求

该方法只要求输入股票代号以及图的类型即可显示出该股票的发展曲线图。在图的类型中用 D 代表日、W 代表周、M 代表月。

输入参数的界面如图 26.12 所示。

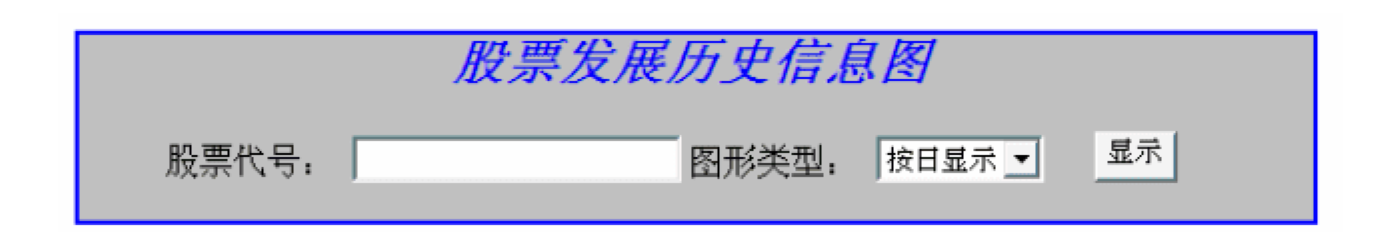


图 26.12　用图形显示股票信息的输入界面

其中图的类型用下拉列表中的 SelectedValue 值来表示。【显示】按钮的代码按照下面的要求输入即可。

```
protected void Button1_Click(object sender, EventArgs e)
 {
  string stockCode = TextBox1.Text;
                  //取出股票代号
 string ImageType = DropDownList1.SelectedValue;
                 //取出日、周或月图形类型
HttpContext.Current.Response.Cache.SetCacheability(System.Web.HttpCachea
bility.NoCache);
 stockimage.ChinaStockWebService proxy2 = new
                  stockimage.ChinaStockWebService();
byte[] pp = proxy2.getStockImage_kByteByCode(stockCode, ImageType);
HttpContext.Current.Response.ClearContent();
 //清除缓冲区流中所有输出，不清除可能图片不能到顶
HttpContext.Current.Response.Buffer = true;
 //缓冲输出，可以不使用
  HttpContext.Current.Response.ContentType = "image/Gif";
 //获取或设置输出流中的 HTTP MIME 类型(文件头类型)
 HttpContext.Current.Response.BinaryWrite(pp);
 //将一个二进制字符串写入 HTTP 中
 HttpContext.Current.Response.End();
 //将所有缓冲区输出发送到客户端，停止该页执行
 }
```

显示股票图形的界面如图 26.13 所示。

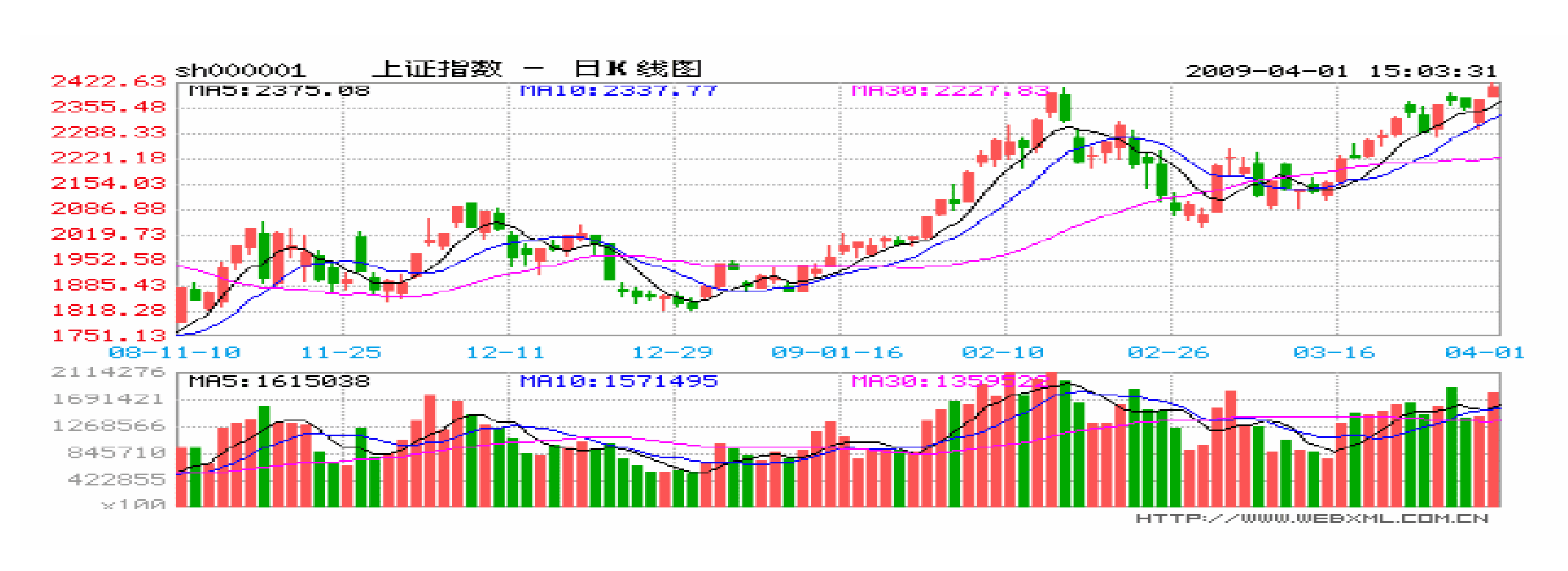


图 26.13　用图形显示股票信息的界面

26.4.4　电视节目预报信息

电视节目预报信息的设计比前面几项设计要稍为复杂一些。因为每个地区有多个电视台；各个电视台有多个频道；每个频道有很多不同的节目。显示结果如图 26.14 所示。



图 26.14　电视节目预报信息界面

在网站中创建代理与前面几项相同。现在选择【中国电视节目预告(电视节目表)】，找到服务类为 ChinaTVprogramWebSercice.asmx。需要用到的方法有如下几个。

- getAreaDataSet()：查找地区或分类的电视的列表。
- getTVstationDataSet()：根据省、市 ID 或分类电视 ID 获得电视台列表。
- getTVchanelDataSet()：根据电视台 ID 获得电视频道列表。
- getTVprogramDataSet()：根据频道 ID 和日期获得节目预报列表。

下面设计的代码分三段分别执行以下三项工作。

第一段：打开网页时(Page_Load 事件)将自动获得各省市 ID 及分类电视 ID 列表，并放置于地区的下拉菜单中。

第二段：从选择地区的下拉列表中选择地区后，单击【按地区查找电台】按钮(Button1_Click 事件)，将查到的电视台动态地加入到电视台的下拉列表中。

第三段：在选择电视台的下拉列表中选择电视台名，并单击【预告节目】按钮(Button2_Click 事件)时，界面中以表格的形式显示节目预报。

以上各段工作相应的代码如下。

```
public partial class 电视节目预告: System.Web.UI.Page
{
 string se;
 protected void Page_Load(object sender, EventArgs e)
 {
  //第一段工作
  TVprogram.ChinaTVprogramWebService t = new
               TVprogram.ChinaTVprogramWebService();
  DataSet d = new DataSet();
  d = t.getAreaDataSet();
  DataTable DA = d.Tables[0];
     int row = DA.Rows.Count;
     int i;
     if (DropDownList1.Items.Count == 0)
      {
       for (i = 0; i < row; i++)
       {
        DropDownList1.Items.Add(new
        ListItem(DA.Rows[i].ItemArray[1].ToString(),
        DA.Rows[i].ItemArray[0].ToString()));
        }
       }
```

```
            se = DropDownList1.SelectedValue;
          }

    protected void Button1_Click(object sender, EventArgs e)
{
  //第二段工作
     se = DropDownList1.SelectedValue;
     DropDownList2.Items.Clear();
     TVprogram.ChinaTVprogramWebService t = new
     TVprogram.ChinaTVprogramWebService();
     DataSet d = new DataSet();
     d = t.getAreaDataSet();
     DataTable DA = d.Tables[0];
     int row = DA.Rows.Count;
     int i;
     d = t.getTVstationDataSet(int.Parse(se));
     DataTable DA1 = d.Tables[0];
     string temp;
     for (i = 0; i < DA1.Rows.Count; i++)
     {
      temp=DA1.Rows[i].ItemArray[0].ToString();
      d = t.getTVchannelDataSet(int.Parse(temp));
      DataTable DA2= d.Tables[0];
      int j;
      for (j = 0; j< DA2.Rows.Count; j++)
      {
        string chanenelid;
        chanenelid = DA2.Rows[j].ItemArray[0].ToString();
        DropDownList2.Items.Add(new ListItem(DA2.Rows[j].ItemArray[1].ToString(),
                 DA2.Rows[j].ItemArray[0].ToString()));
          }
         }
}

    protected void Button2_Click(object sender, EventArgs e)
{
     //第三段工作
        if (DropDownList2.Items.Count==0)
        { Response.Write("<script Language='JavaScript'>alert('抱歉还没有选
        择电视台！！！')</script>"); return; }
        TVprogram.ChinaTVprogramWebService t = new
            TVprogram.ChinaTVprogramWebService();
        DataSet d = new DataSet();
        d = t.getTVprogramDateSet(int.Parse(DropDownList2.SelectedValue),
        "", "");
        DataTable DA3 = d.Tables[0];
        GridView1.DataSource = DA3;
        DataBind();
        if (GridView1.Rows.Count == 0)
            Response.Write("<script Language='JavaScript'>alert('抱歉还没有
                                预告！！！')</script>");
       }
}
```

26.5 创建 XML Web 服务网站

前面已经讲述了作为消费者调用服务的方法，在这里将讲述创建 XML Web 服务网站的方法。XML Web 服务可以用.NET 系统创建，也可以在 J2EE 或其他开发平台上创建。下面以创建一个温度转换的 Web 服务为例，讲述利用.NET 框架的 ASP.NET 平台创建 XML Web Service 服务网站的方法。

26.5.1 创建.asmx 文件

由于.NET Framework(框架)已经为 XML Web 服务网站建立了基础框架，因此利用 ASP.NET 创建 XML Web 服务与创建一般网站的方法区别不大，关键在于创建好.asmx 文件。

.asmx 是一个用于服务的类文件，它与一般网页文件(.aspx 文件)的基础不同，但从文件的结构上看区别不大。主要区别在于：

- 用于服务的类(.asmx)继承于 System.Web.Services.WebService，而不是继承于 Page 类。因此一开始它就具备 Web 服务的基本功能。
- .asmx 文件只包含逻辑处理代码，也可以用来访问数据库，但不能使用可视化控件，或进行显示界面的设计。
- 为了允许消费者调用，.asmx 文件中必须包括有用 public 定义，并且有一定返回类型的方法。
- 文件中定义了若干属性(attribute)。这些属性将自动编译成 WSDL 解释文件，以便向消费者公开调用的接口。

26.5.2 创建温度转换的 Web 服务

由于 Web 服务必须通过 IIS 信息服务器进行发布，因此在系统中必须首先安装好 IIS 服务器，并且通过虚拟目录来发布 Web 服务。为此，可以直接将 Web 服务文件放置到虚拟目录(或其子目录)下，也可以先将 Web 服务文件放在任一个物理目录下，然后用虚拟目录指向对应的物理目录。下面将采用后一种方式创建 Web 服务。

1. 创建的步骤

创建 XML Web 服务的步骤如下。

(1) 选择【文件】|【新建网站】命令。

(2) 在打开的对话框中单击【ASP.NET Web 服务】按钮。

(3) 给网站取名(这里取名 Temperate)，选择使用 C# 语言以及存放地址(可以选择任意物理地址)。

(4) 打开网站目录，将看见网站的根目录下出现了 Service.asmx 以及在 App_Code 目录下的 Service.cs 两个文件。前者用于显示和检验服务文件，后者存放实际的代码。

(5) 单击 Service.cs 文件后将看到该文件的初始代码如下。

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;

[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
// 若要允许使用 ASP.NET Ajax 从脚本中调用此 Web 服务，取消对以下行的注释
// [System.Web.Script.Services.ScriptService]
public class Service : System.Web.Services.WebService
{
  public Service () {
  //如果使用设计的组件，取消注释以下行
  //InitializeComponent();
}

[WebMethod]
public string HelloWorld() {
    return "Hello World";
  }
}
```

注意这个文件与.ASPX 网页的区别：类的基类是 WebService；代码中增加了几个用方括号括起来的属性(attribute)。如[WebService(…)] 、[WebMethod]等，这些都是.asmx 文件中必须的。

其中用 public 定义的方法就是提供给消费者调用的方法。现在将其改写如下。

```
    [WebMethod()]
    public double FahrenheitToCelsius(double Fahrenheit)
    {
        //华氏转换为摄氏
          return ((Fahrenheit - 32) * 5) / 9;
    }
    [WebMethod()]
    public double CelsiusToFahrenheit(double Celsius)
    {
        //摄氏转换为华氏
        return ((Celsius * 9) / 5) + 32;
}
```

(6)　指定.asmx 文件作为起始页，进行试运行。正常时弹出的窗口中将出现两行字符串，这就是.asmx 文件中提供服务的两个方法名，单击它们就可以看出运行的结果。

- CelsiusToFahrenheit
- FahrenheitToCelsius

假定在输入窗口中输入 34，然后单击第一项(CelsiusToFahrenheit)，再单击【调用】按钮，将显示转换的结果(93.2)，如图 26.15 所示。

服务的结果用 XML 的形式显示。Web XML 服务并不提供界面设计的服务，因此只能用来验证结果是否正确。



图 26.15　初步显示转换结果

2. 在本机上验证 XML Web 服务

可以在本机上自己设计界面来进一步验证服务的结果。

如前所述，一台计算机可以接受来自第三方的服务。其实计算机也可以接受本机的服务。这就是说，一台计算机的浏览器与本机的服务器之间也可以采用消费方与服务方之间的通信方式。例如客户认证或某种计算，就可以用提供服务的方式来实现。为了与不同机器之间的服务一致，接受本机服务时仍然需要设置代理(Proxy)。

下面将在本机上调用上面设计的 XML Web 服务，来检查服务的效果。具体步骤如下。

(1) 创建并注册代理(Proxy)。

① 在网站菜单上单击【添加 Web 引用】按钮，将出现如图 26.16 所示的对话框。



图 26.16　注册代理

② 选择【本地计算机上的 Web 服务】选项。

③ 后面的操作均与第 25 章相同，这里不再重复。

(2) 增添新网页，设计新界面，如图 26.17 所示。

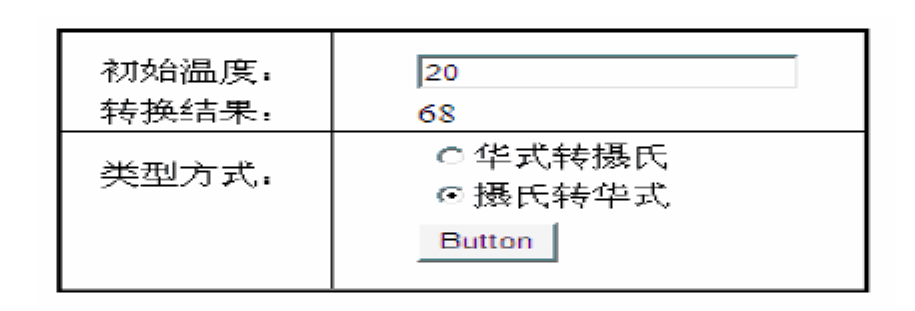


图 26.17　温度转换显示界面

(3) 为按钮(Button)的 Click 事件编写代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
{
 //生成代理对象
```

```
  Temperature.Service proxy1 = new Temperature.Service();
  double tt = double.Parse(TextBox1.Text);
  //显示结果
if (RadioButtonList1.SelectedValue == "0")
{
    Label1.Text=proxy1.FahrenheitToCelsius(tt).ToString();
}
else
{
    Label1.Text=proxy1.CelsiusToFahrenheit(tt).ToString();
 }
}
```

26.6　小　结

Web 服务是机器之间的对话，它突破了系统、平台、语言等的界限，把关系建立在高度松耦合的基础之上，实现服务的必要条件是双方必须遵守因特网通用的通信标准和服务与消费之间的协议。

为了简化服务的过程，系统在消费方建立了代理，它是服务方驻留在消费端的代表，并在消费方进行了注册。消费方通过代理访问服务端，就如同访问内存的对象那样简单。

调用服务的程序设计时，先要找准服务的网站，确定服务类的 URL，看清楚服务项目的接口说明文件。该文件通常比较长，关键是要看清楚服务的类名(.asmx)、调用的方法名，以及调用时需要提供的参数和返回数据的类型等几方面的问题。返回的数据类型可能是数据、字符串、一维字符串数组、数据集、某种图形或者它们的混合。如何处理返回的数据本章的典型示例中均有说明。

因为.NET Framework 已经提供了创建 XML Web 服务网站的基础框架，创建 XML Web 服务网站并不难，与创建传统网站时的步骤差别不大。应注意创建.asmx 类文件与.aspx 网页的不同点，主要修改框架中提供的方法，并且在创建完成后进行验证。

26.7　习　题

1. 填空题

(1) Web 服务建立在网站之间__________________的基础之上。
(2) 建立服务方与消费方关系的唯一基础是_______________与_______________。
(3) SOAP 是英文_____________________________的缩写。
(4) XML Web 服务方提供的服务主要包括__________和__________两方面。
(5) 作为服务方代理必须在消费方_____________。
(6) WSDL 文件是用_________的格式来描述________________________。

2. 选择题

(1) 在 XML Web 服务中，代理是________的代表。

A. 服务方　B. 消费方　C. 客户方　D. A + B

(2) 用 SOAP 来传输________时最合适。

A. 字符串　B. XML　C. 命令或指令　D. B + C

3. 判断题

(1) 一个 XML Web 的服务方不能又是消费方。 (　)

(2) XML Web 服务受开发平台的限制。 (　)

(3) XML Web 服务不受使用语言的限制。 (　)

(4) 计算机不能接受来自本机的服务。 (　)

(5) 计算机接受本机服务时不需要设置代理。 (　)

4. 简答题

(1) .asmx 是什么文件？它与.aspx 文件有什么不同？

(2) 在.asmx 文件中的属性的作用是什么？

5. 操作题

(1) 设计一个网站调用气象、航班和股票服务。

(2) 设计一个网站为电视台节目预报服务。

(3) 设计一个网站通过自行查阅文档调用 2～3 项信息服务。

(4) 创建一个简单的 XML Web 服务网站。

第七部分

综合示例

一套实际应用的系统中常常包括比较复杂的内容，既有核心代码，还有千变万化的外围设计。作为一名初学者，应该首先掌握关键知识与核心技术，并在这个基础上进一步扩展和完善设计。

在下面的讲解中，将着重讲述一些关键技术。读者只有在理解和掌握这些知识的基础上，再加上适当的加工才能应用于实际之中。

本部分选择了三种不同类型的主题：

- 网上招聘与留言板。
- 快速创建动态数据驱动网站。
- 创建电子商务网站。

第 27 章　网上招聘与留言板

由于网站工作在 Internet 开放的环境中，因此网站可以在最广阔的范围内交换信息。如果应用于网上报名、网上招聘、民意咨询、留言板等项目时，将给组织者和客户带来极大的方便。它不受时间、地点的限制，不仅效率高，还能降低工作成本，因此用它来取代传统方式已经成为大势所趋。

本章将讲解两个应用项目：网上招聘与留言板。学会了这两种项目的设计，其他类似的项目都可以迎刃而解。

在本章中将利用 FormView 控件来设计，需要用到前面几章中学过的内容，所以它是一个综合性的章节。本章中将要讲解的具体问题包括:

- FormView 控件简介。
- 利用 FormView 控件设计招聘网页。
- 利用 FormView 控件设计留言板。
- 使用 Wizard 控件。

27.1　FormView 控件简介

一个信息交换的应用程序应该同时具备以下三方面的功能。

- 给客户提供一个友好的输入界面。
- 能够自动保存客户输入的数据。
- 能够对输入的数据进行整理、分析和处理。

其中“自动保存客户输入的数据”是设计的关键，也是用传统方法设计的难点。现在 ASP.NET 3.5 对这个问题提供了强有力的支持，大大简化了设计过程。下面采用 FormView 控件并结合数据库来进行设计。

FormView 控件与前面讲过的 GridView 控件都继承于 CompositeDataBoundControl 类，因此有很多共性。它们都可以用来显示数据表，并且对数据表进行编辑。现在利用它拥有的“插入记录”的功能来自动保存客户输入的数据。

FormView 与 GridView 控件之间最重要的区别是，FormView 控件的显示界面可以更加自由，而不限于表格形式，因此更容易适合于不同显示界面的需要。

27.2　利用 FormView 控件设计招聘网页

一个招聘广告通常需要先介绍招聘单位本身的简要情况，并提出招聘的要求，然后给应聘者提供一个登记的界面。该界面应该尽量友好，使客户能够用最简便的方式输入需要的数据，为以后进一步分析和联系奠定基础。

例如，图 27.1 是一个“招聘软件开发人员”的网页。

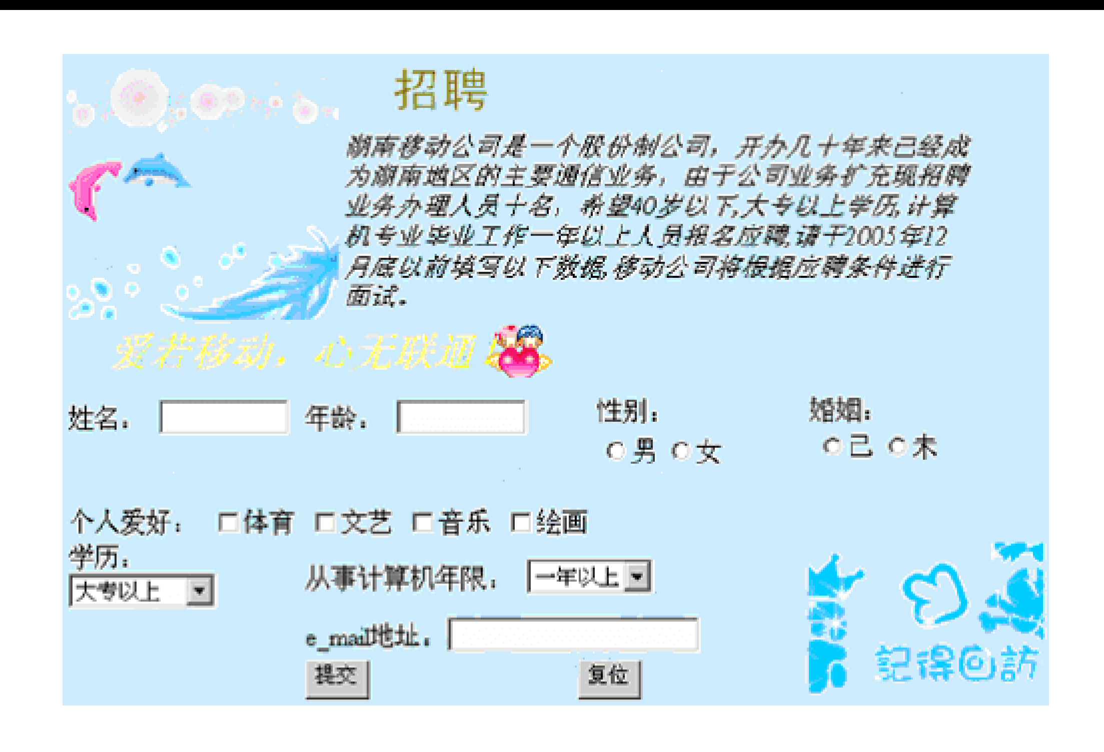


图 27.1　招聘网页的界面

网页的界面分为上、下两部分。上面部分使用 FormView 控件的 Header 模板介绍公司情况并提出招聘要求。下面部分使用 Insert 模板，放入各类输入控件并与数据表各字段进行数据绑定，以便自动保存输入的数据。具体设计步骤如下。

(1) 创建一个数据表，数据表的字段就是需要了解的项目。以前面的招聘为例，数据表包括 12 个字段，其格式定义如表 27.1 所示。

表 27.1　定义数据表字段

字 段 名	类　型	字 段 名	类　型
Bh(编号)	int 4	Name(姓名)	nvarchar 12
Sex(性别)	nvarchar 4	Age(年龄)	int 4
Marriage(婚姻状况)	char 10	Sport(爱好体育)	char 10
Art(爱好艺术)	char 10	Music(爱好音乐)	char 10
Drawing(爱好绘画)	char 10	Schooling(学历)	nvarchar 20
Experience(资历)	nvarchar 12	Email(电子邮件)	nvarchar 30

其中，编号(bh)是关键字，设为自动增量，因此不出现在输入的界面中。

(2) 将 FormView 控件与数据库连接。从工具箱将 FormView 控件拖入窗体，并通过数据源控件与数据库连接。注意在连接过程中单击【高级】按钮，以便生成各种 SQL 编辑语句。

(3) 创建输入界面。FormView 控件提供了多个模板，分别用于不同需要时的界面设计。如项目模板(ItemTemplate)、编辑模板(EditItemTemplate)、插入模板(InsertItemTemplate)、头模板(HeaderTemplate)和尾模板(FooterTemplate)等。

此例使用 FormView 控件的插入模板。

方法是：右击 FormView 控件，在弹出的菜单中选择【编辑模板】| InsertItemTemplate 命令，即可转向插入模板。当模板出现时，其中已经包含了一些控件，只是这些控件的布局并不一定符合设计要求，可以先删除这些控件。

接着给模板进行布局，然后放入各种类型的输入控件(如 TextBox、DropDownList、RadioButtonList、CheckBox 等)。除此以外还需放入两个按钮：【保存】和【复位】按钮。在这些控件中除【复位】按钮要采用 HTML 的 Reset 按钮以外，其他控件均采用服务

器端标准控件(Label 控件也可以采用 HTML 控件)。

(4) 进行数据绑定并设置其他参数。通过各控件的【编辑 DataBindings】属性分别与数据表对应的字段进行数据绑定，如图 27.2 所示。

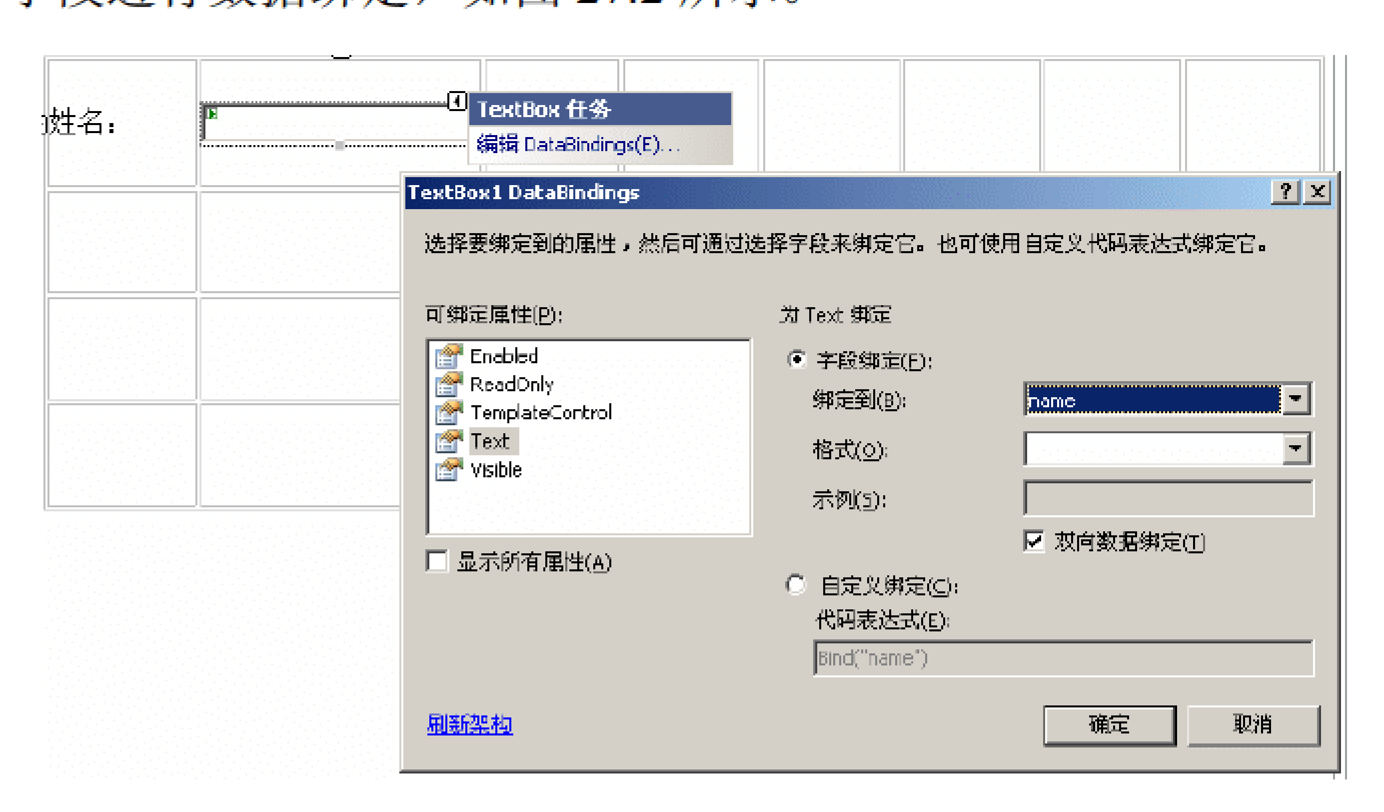


图 27.2 给各控件进行数据绑定

图中表示 TextBox1 控件的 Text 属性已经与数据表的 name 字段进行绑定。窗体下面的代码表达式：Bind("name")就表明了这种绑定关系。

① 给【保存】按钮的 CommandName 属性赋值为 insert。

② 给各输入控件增加相应的校验控件。

(5) 为了使得打开网页时立即打开插入模板，应将插入模板设置为 FormView 的默认模式。为此，先退出插入模板，然后在 FormView 控件中，将 DefaultMode 属性设为 Insert。

(6) 打开头模板(HeaderTemplate)，在头模板中介绍公司情况并提出招聘要求。

(7) 给应聘者返回信息。方法是：在 FormView 控件的 ItemInserted 事件中输出信息。例如编写有关感谢的语句如下。

```
void FormView1_ItemInserted(object sender, FormViewInsertedEventArgs e)
    {
        Response.Write("感谢您参加应聘！…");
    }
```

27.3 利用 FormView 控件设计留言板

留言板可以为客户提供网上留言的机会，它不受时间、地点的限制，是及时获取客户反馈信息的一种好方式。

网站的留言板应该给客户提供一个简单、友好的输入界面；还要能自动保存这些信息并允许客户随时查看其他客户的留言。

留言是一种面向社会开放的机制，因此也难免会出现一些过时的或不健康的留言，应该允许网站管理人员定期进行清理。

因此设计留言板网页要综合应用增添记录、校验输入、删除留言、登录检验等技术。利用系统提供的控件，可以很方便地实现这些功能。

27.3.1　打开留言板

图 27.3 是一个简单的留言板界面。界面中只包括三个链接指针：开始留言、查看留言与留言管理，通过它们打开留言区网页、查看网页以及管理留言网页。



图 27.3　留言板开始界面

27.3.2　留言网页的界面设计

留言网页的设计与招聘网页的设计步骤基本相同，这里只重点讲述几点不同之处。

- 为了保存客户的留言，须先建立一张数据表，包括留言编号、姓名、主题、电子邮件、留言时间以及留言内容等几个字段，其中留言编号是关键字(不能重复)，在数据表中用自动增量方式生成，但不一定显示在界面上。数据表的结构如表 27.2 所示。

表 27.2　留言网页所需字段

字 段 名	类　型	字 段 名	类　型	字 段 名	类　型
Bh(编号)	int 4	ZT (主题)	varchar 40	SJ (时间)	varchar 50
XM (姓名)	varchar 12	E-mail(电子邮件)	varchar 30	NR (内容)	varchar 800

其中，Bh 为关键字，自动增值；SJ(时间)将由系统自动生成，这里采用 varchar 类型。

- 拖入 FormView 控件，然后按照前面步骤进行设置。其中留言内容的 TextBox 文本框的 Type 的属性设成多行，并将它的 MaxLength 属性设置成最多允许字长(例如 800)。留言板的界面如图 27.4 所示。
- 除 SJ(时间)文本框以外其他均与数据表进行数据绑定。
- 在 SJ(时间)文本框中按照以下方式进行数据绑定。即先选择【自定义绑定】(Custom Binding)，然后再与系统时间(DateTime.Now)绑定在一起。同时将时间的文本框的 ReadOnly 属性设为 True，如图 27.5 所示。
- 利用 FormView1_ItemInserting 事件将系统时间存入数据表中。语句如下。

```
void FormView1_ItemInserting(object sender, FormViewInsertEventArgs e)
  {
```

```
            SqlDataSource1.InsertParameters.Clear();
            SqlDataSource1.InsertParameters.Add("sj", DateTime.Now.ToString());
        }
```



图 27.4　留言板界面

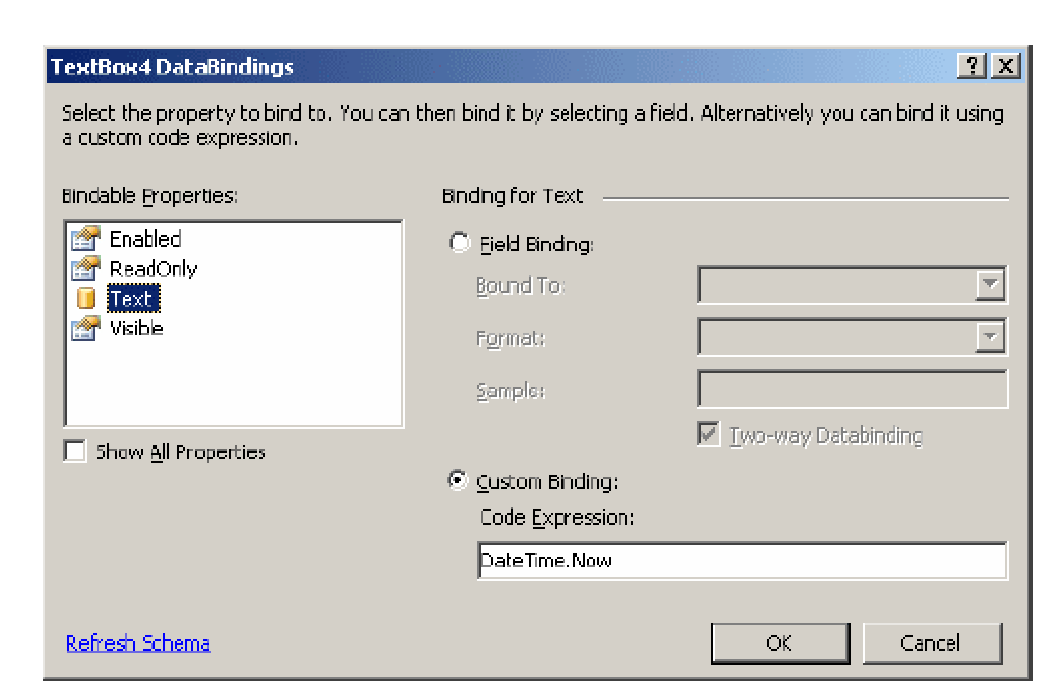


图 27.5　自动显示留言时间

- 在 FormView 控件的 ItemInserted 事件中输出语句表示已经收到留言，并表示感谢。语句如下。

```
void FormView1_ItemInserted(object sender, FormViewInsertedEventArgs e)
  {
    Response.Write("感谢您的留言!");
  }
```

上述语句也可以改写成以下形式。

```
Response.Write("<script Language='JavaScript'
type='text/javascript'>alert('感谢您的留言！'); </script>");
```

后面这条语句的作用是向浏览器发送一个字符串，在浏览器将其下载后自动编译成 JavaScript 脚本，显示出一个感谢留言的小对话框。

27.3.3　对留言板的管理

网站管理人员应该定期或不定期地对留言板进行清理，及时删除一些过时的信息或者一些信息垃圾。为此必须为管理留言板的工作设计一些网页，打开这些网页的人应该具有一定的权限。关于设置权限的方法在前面的章节中已经讲述。

清理留言板中的记录可以利用 GridView 控件与留言板的数据表连接，并且按照编辑方法配置数据源控件，并在 GridView 控件中增加【删除】按钮，以便有选择地删除留言。

27.4　使用 Wizard 控件

27.4.1　Wizard 控件的用途

Wizard 控件的作用是将多个窗口放在同一个网页的不同模板中，然后利用显示/隐含模板的方法，在界面之间进行切换。就像分页显示一样。但与分页不同的是，Wizard 控

件中放进的是独立而又相关的界面。例如一个企业的人员组成、产品销售、财务情况等就可以分别放入不同的模板中。前面讲述的留言板如果有多种界面时，也可以将它们放入不同的模板中。

在 Wizard 控件中，这些模板将按照步骤(第一步、第二步等)进行排列，客户可以按顺序来逐个浏览界面，也可以用跳转的方式进行查阅。由于界面的切换都是在同一张网页的各个模板中进行，因此切换的速度比较快，使用起来非常方便。

27.4.2 Wizard 控件的结构

Wizard 控件的界面分为 4 部分，如图 27.6 所示。

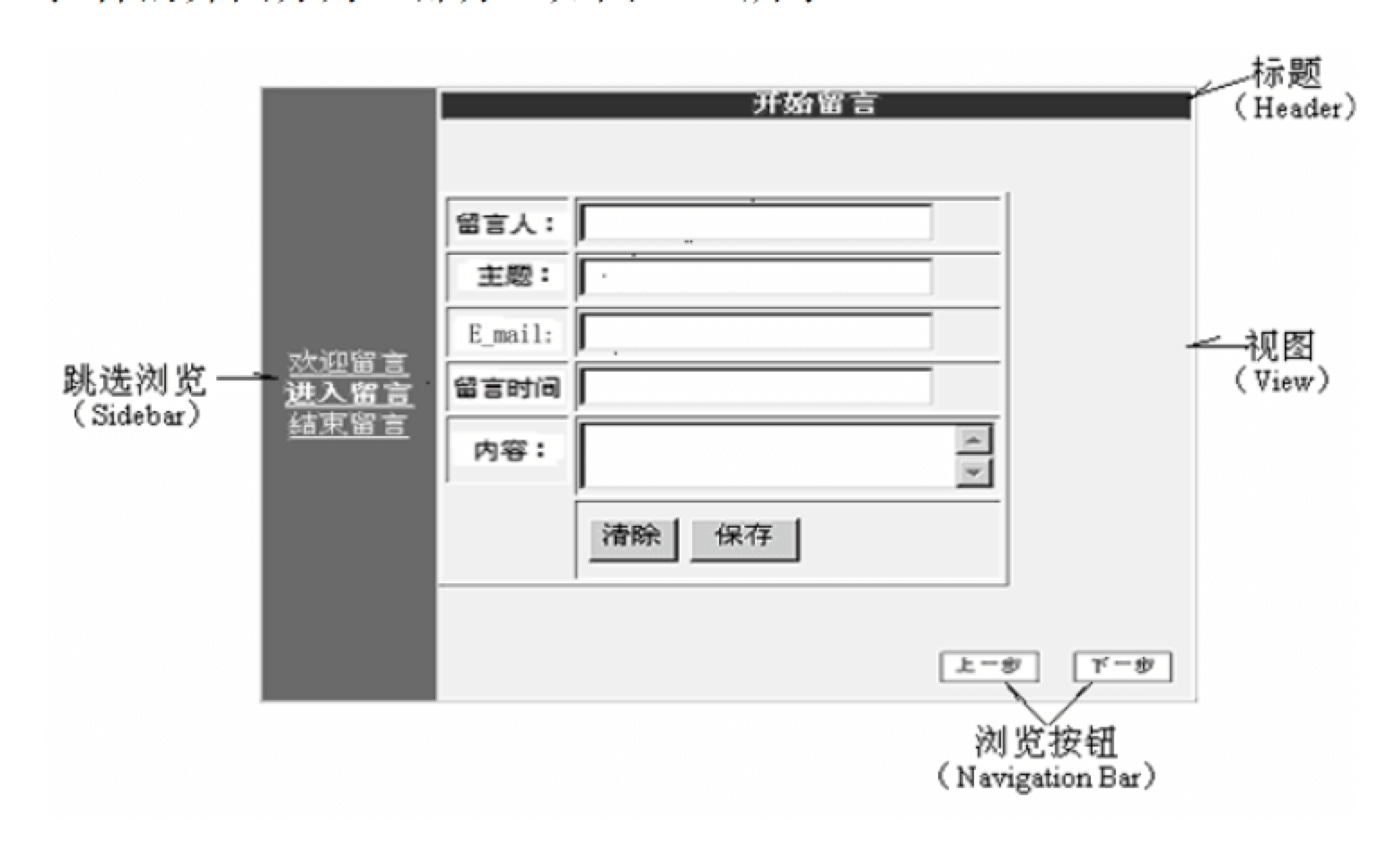


图 27.6 Wizard 控件的结构

- 标题部分(Header)：放在界面的上方，起标题的作用。
- 视图部分(View)：界面的主要部分。中间放入各种控件组成需要的界面。
- 浏览按钮部分(Navigation Bar)：通常放在界面的下方，由【上一步】、【下一步】等按钮组成，用来执行逐步浏览的功能。
- 跳选浏览部分(Sidebar)：通常放在界面的左方，利用它可以采用跳选方式浏览界面。这一部分是可选的部分。只要将 Wizard 控件的 DisplaySideBar 属性设为 false，该部分就被隐藏起来。

27.4.3 Wizard 控件的使用方法

1. 设置浏览的步骤

将 Wizard 控件拖进窗体，单击属性 Wizard Steps 右边的省略号按钮，弹出如图 27.7 所示的对话框。

利用【添加】按钮结合右边的 Title 属性，给 Wizard 控件增添步骤，并给各个步骤命名。

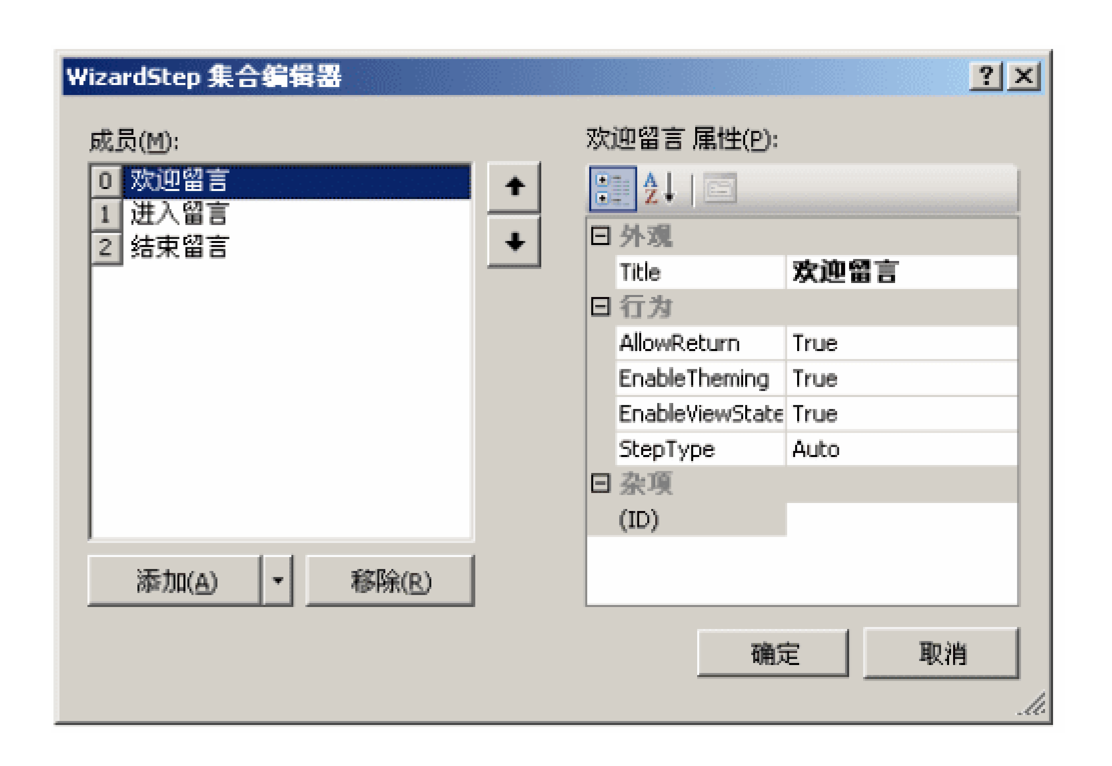


图 27.7 【WizardStep 集合编辑器】对话框

2. 给各模板设置属性

单击左边窗口中的步骤，分别为每步的界面增添需要的控件，增添控件的方法与在网页中增添控件一样。除此以外，在 Wizard 控件的【属性】对话框中，系统为各个模板以及模板中的按钮提供了丰富的属性，如底色(BackColor)、边界样式(BorderStyle)、边界线宽度(BorderWidth)、边界颜色(BorderColor)等。可以按照自己的爱好来设置这些属性。一个简便的方法是先选择【自动套用格式】选项，在系统提供的多种格式中先选择一种，然后再分别给各个模板做进一步的修改。

除此之外，Wizard 控件还提供了十几个事件，必要时可以在事件窗口中双击某个事件，并为该事件编写代码。具体做法与一般网页相同，这里就不介绍了。

27.5 小　结

招聘网页和留言板是使用得非常广泛的两个典型项目，它们的主要功能都是给客户提供一个友好的界面，要求客户提供相关数据，自动保存这些数据，以便进行分析和处理。两者之间只是显示的界面不同。使用 FormView 控件来做这些项目非常合适。因为这个控件能够通过数据源控件连接数据库，自动执行插入记录的操作。另外，由于 FormView 控件的显示界面比较自由，可以灵活地进行布局。控件还提供了多种模板，可以根据需要进行选择。设计中需注意三件事：

- 设计出一个友好的界面。
- 各个输入控件与数据库之间正确地进行数据绑定。
- 对输入的数据要进行必要的校验。

Wizard 控件是 ASP.NET 3.5 提供的一个控件，这个控件的特点是能够将几个独立而又相关的窗体放到同一个网页中来，通过浏览按钮或跳转选择可以在这些网页间进行浏览，由于这种方法设计简单，运行中切换的速度也比较快，因此比较受欢迎。

27.6　习　　题

1. 填空题

(1) 网上招聘模板通常分为上、下两部分。上面部分通常使用 FormView 控件的__________，用来介绍公司情况并提出招聘要求；下面部分通常使用___________，用来输入和存储数据。

(2) 当应聘者提交数据后应该在___________事件中向应聘者发出友好的信息。

(3) 通常在应聘的界面上要放入【保存】和【复位】两个按钮，其中【复位】按钮采用__________控件，而【保存】按钮则必须是__________控件。

(4) 如果要求自动保存提交数据的时间，应该在____________事件中编写保存当前时间的代码。

2. 选择题

(1) FormView 与 GridView 控件相比最重要的区别是_________。

A. 能够存储数据　　B. 外观比较美观

C. 显示的布局几乎不受限制　　D. 数据量受一定的限制

(2) 留言板必须加强管理，这是因为____________。

A. 存储容量有限　　B. 查看必须经过授权

C. 影响查看的速度　　D. 可能存在过时或不健康的留言

(3) Wizard 控件的最大特点是__________。

A. 能够包含大量数据　　B. 能将多个窗口集中到一个网页中

C. 布局不受限制　　D. 能够快速进行浏览

3. 简答题

(1) 一个网上信息交换程序通常应该包括哪几方面的功能？

(2) 简述网上招聘网页的设计步骤。

(3) 简述设计留言板的设计步骤。

(4) Wizard 控件的作用是什么？

4. 操作题

(1) 实际设计一个研究生网上报名的网页，要求对输入的数据进行合法性验证，并能自动保存数据(包括提交时间)，数据提交后能够向报名者反馈相关信息。

(2) 利用存储过程(不用 FormView 控件)完成上题的要求。

(3) 设计一个网上的舆论调查程序，要求在客户提交自己的意见以后立即显示到目前为止调查的综合结果。

(4) 利用 Wizard 控件设计一个包括进入留言、实际留言、管理留言等多页的应用程序。

第 28 章　快速创建动态数据驱动网站

相当一部分应用程序以数据库为基础，以数据处理为中心，如何快速建立起数据处理的模型，已经成为设计者普遍关心的问题。

ASP.NET 3.5 提供了 LINQ 集成语言查询，并且通过 LINQ to SQL 将数据库(数据表)映射成实体类，使得对数据的处理更加方便和安全，并为快速创建系统提供了坚实的基础。除此之外，系统还提供了一整套基于 LINQ 语言的动态数据网站的创建方法。

本章将以样板库 Northwind 为基础讲述动态数据驱动网站的创建方法。

28.1　概　　述

房屋建筑设计可以采用两种方式，一种就是根据各方面的条件(需要、资金、地形、环境等)，从头开始进行设计，设计的针对性强，能够满足各种不同的需要。但是这种设计方式成本高、周期长。

另一种方式(例如经济适用房)是采用标准化设计方式。即先根据普遍的需要设计几套方案，需要时选择其中之一(或许需要做少量的修改)即可开始施工。这种设计成本低、周期短，但是设计的结果缺乏个性，不能完全满足不同的需要。

软件设计与房屋建筑设计性质不同，但也有相似的地方。前面讲述的网站设计实质上都是采用第一种设计方式。现在 LINQ 技术的出现为第二种设计方式提供了可能。

LINQ 不仅是一种新技术，它也扩展了人们的想象空间。人们在想，既然通过 LINQ 能够将数据库映射成实体类，为什么不能直接将数据库映射成软件系统？在这种思维的启发下，快速创建动态数据驱动网站的技术诞生了。

与传统创建网站的方式不同，创建动态数据驱动网站时，先根据需要组成数据库，然后在此基础上快速创建一个数据驱动系统，这个系统具有基本的显示界面和操作功能，包括显示、过滤、分页、排序、添加、删除、编辑、修改以及输入校验等通用的功能。创建这个系统的过程几乎不需要编写代码，也不需要设计人员做什么干预就能够自动完成。

创建过程非常快，是这个系统的最大特点。但由于系统是按照通用模型生成的，还缺乏个性，需要设计者与使用者紧密配合进行补充和完善。由于已经有了一个可以实际运行的基本模型，使用者提意见时将更有针对性，设计者也更容易理解和实施，两者之间的配合将更加默契。

28.2　数据模型(Model)与支架(Scaffold)

系统提供的数据模型和支架是创建数据驱动网站的基础。

什么是数据模型？数据模型就相当于标准设计的方案，在这里它代表数据库中各信息(字段)之间的相互关系。比如：是各种表格都支持统一的支架，还是各个表分别支持不同

的支架等。目前，创建数据模型有两种方式：LINQ-to-SQL 和 ADO.NET Entity Framework。本章将重点介绍前者。

什么是支架(Scaffold)？支架这个名称来源于建筑行业，它是设计方案的细节和具体化。在这里它的作用是根据数据模型，自动生成网页外观，并赋予系统各项基本功能。

选择数据模型和支架的过程非常简单，只要在 Global.asax 文件中释放几个原来被屏蔽的定义字段，或者屏蔽另外几个字段即可完成。

28.3 创 建 步 骤

下面以 Northwind 样板数据库为例，讲解创建数据驱动网站的方法。

(1) 选择【文件】|【新建网站】命令。

(2) 在打开的模板对话框中选择【Dynamic Data 网站】。

(3) 给网站取名，选择【文件系统】网站，并确定使用 C#语言。

(4) 在 App_Data 目录下引入 Northwind 数据库。

(5) 选择【添加新项】，并选择【LINQ to SQL 类】选项，以打开 O/R 设计界面，并将文件改名为 Northwind.dbml。

(6) 将 Northwind 数据库中的若干相关的数据表拖入到 O/R 设计界面中。

注意： 如果在 Data 选项中选择 ADO.NET Entity Data Model 时，应按照以下步骤进行。

(1) 在 Name 小框中取名，如 Northwind.edmx，单击【下一步】按钮。

(2) 选择【从数据库生成】小图标，然后单击【下一步】按钮。

(3) 在【您的应用程序将使用哪个数据连接连接到数据库？】下面的小窗口中选择 NorthwindConnectionString(Setting)。

(4) 保存上面的设置，单击【下一步】按钮。

(5) 选中要用到的数据表，然后单击【完成】按钮。

(7) 打开 Global.asax 文件以选择数据模型。

在传统的网站中，Global.asax 文件用来控制一些全局性的事件，每当应用程序启动时将首先执行此文件中的代码。在动态数据网站中，还需要在此文件中进行“数据模型注册”。

数据模型分为两种。

- 分页模式，即列表、详细、插入和更新任务放在不同的网页中执行。
- 合页模式，即列表、详细、插入和更新任务都放在同一网页中执行。

Global.asax.文件的提示如下(注意其中用粗字体显示的代码)。

```
public static void RegisterRoutes(RouteCollection routes)
 {
    MetaModel model = new MetaModel();
   //                  重要: 数据模型注册
   // 取消注释此行以注册 LINQ to SQL 类或 ASP.NET 实体数据的
   // ADO.NET 动态数据模型。若要设置 ScaffoldAllTables = true，需符合以下条件
   // 即确定希望数据模型中的所有表都支持支架(即模板)
   // 视图。若要控制各个表的支架，为表创建分部类，并将
```

```
    // [Scaffold(true)] 属性应用于分部类
    // 注意:确保将 YourDataContextType 更改为应用程序的数据上下文类的
    //名称
    // model.RegisterContext(typeof(YourDataContextType), new
                ContextConfiguration() { ScaffoldAllTables = false });

    // 下面的语句支持分页模式，在这种模式下，"列表"、"详细"、"插入"
    //和"更新"任务是使用不同页执行的。若要启用此模式，请取消注释下面
    //的 route 定义，并注释掉后面的合并页模式部分中的 route 定义
   routes.Add(new DynamicDataRoute("{table}/{action}.aspx")
   {
     Constraints = new RouteValueDictionary(new { action =
 "List|Details|Edit|Insert" }),
                Model = model
    });
```

> 注意：如果使用 LINQ-to-SQL 时，上面的 YourDataContextType 用“***DataContext”取代(此例中用的是 NorthwindDataContext)。具体文件名(***)可从***.dbml 下面的***.designer.cs 文件中找到。如果使用 Entity Framework model 时，应该用 Northwind_DataEntities 取代。

```
    //下面的语句支持合并页模式，在这种模式下，“列表”、“详细”、“插入”
    //和“更新”任务是使用同一页执行的。若要启用此模式，取消注释下面
    //的 routes，并注释掉上面的分页模式部分中的 route 定义
    //routes.Add(new DynamicDataRoute("{table}/ListDetails.aspx") {
    //    Action = PageAction.List,
     //    ViewName = "ListDetails",
     //    Model = model
     //});

     //routes.Add(new DynamicDataRoute("{table}/ListDetails.aspx") {
     //    Action = PageAction.Details,
     //    ViewName = "ListDetails",
     //    Model = model
     //});
}
```

如果现在注册“分页模式”则只需要对下列语句进行修改(注意带下划线的三处)。

```
// model.RegisterContext(typeof(YourDataContextType), new
ContextConfiguration() { ScaffoldAllTables = false });
```

修改时：

① 将 “//” 删除。

② 将 YourDataContextType 改为 NorthwindDataContext。

这里的 NorthwindDataContext 是生成.dbml 文件时取的名字并在后面加上 DataContext 类名的结果。这个名字可以从 App_Code 目录下的.designer.cs 文件中找到。

③ 将 false 改为 true。

修改后的语句如下：

```
model.RegisterContext(typeof(NorthwindDataContext), new
             ContextConfiguration() { ScaffoldAllTables = true });
```

到此为止设置完成，整个设置时间大概不超过几分钟。

将 Default.aspx 设为起始页后运行系统。

28.4　系统的目录结构

经过前面的设置，系统将自动生成一个 DynamicData 专用目录，并在这个目录下放置 4 个子目录：Content、CustomPages、FieldTemplates、PageTemplates。

每个子目录下面放置有若干不同类型的文件，这些文件的作用是便于集中修改初始的设置。网站的目录结构如图 28.1 所示。

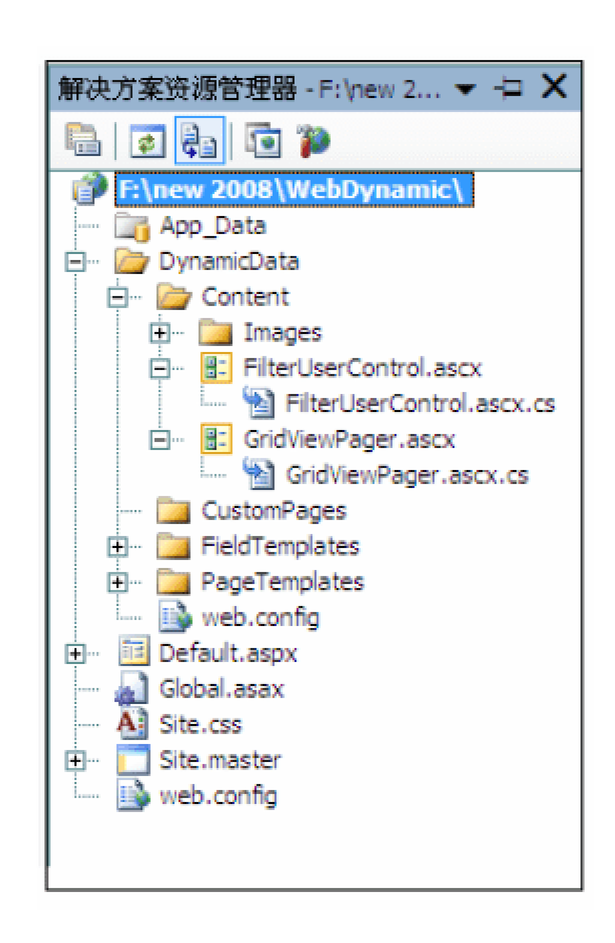


图 28.1　网站的目录结构

28.5　系统的基本功能

经过前面的设置，系统已经具备以下基本功能。

- 通用的显示界面。
- 对数据表进行显示、查询、更新、删除和增添的操作。
- 输入新数据时自动进行验证。
- 各数据表之间利用外键进行连接。
- 对通用界面和功能的快速修改和完善。
- 在网页中自动实现 Ajax 功能。

其分页控制的界面如图 28.2 所示。



图 28.2　分页控制的界面

其多条件组合查询的界面如图 28.3 所示。



图 28.3　多条件组合查询界面

数据表的上面自动出现了多个下拉列表框，以便进行组合查询。数据表中所有的逻辑类型变量都自动用 CheckBox 控件替代。

数据表之间的关系，在 O/R 设计视图中用线条表示，在数据表中用外键字段来表示。如果在数据库中已经建立了多表之间的关系，这些关系将跟随数据表一起带进 O/R 视图或表格中来。在数据表中的关系如图 28.4 所示。如果在数据库中没有建立关系时，也可以利用 O/R 设计视图的工具箱中的“关联”线段来建立关系。



图 28.4　通过外键字段显示连接

通常情况下，数据表的编辑在 GridView 控件中进行，增添新记录的功能在 DetailsView 控件中进行，其界面如图 28.5 所示。



图 28.5　数据表编辑界面

28.6　修改系统的方法

为了便于对生成的结果进行修改，系统提供了集中修改的工具。

(1)　对外观的修改主要通过各自的 CSS 进行。

(2)　对字段的集中修改。系统通过用户控件对数据表中的字段进行定义，并将不同类型的定义文件集中放在 FieldTemplates 目录下。这样做的好处，一是可以集中修改，二是能够保持各网页显示风格一致。FieldTemplates 的目录结构如图 28.6 所示。

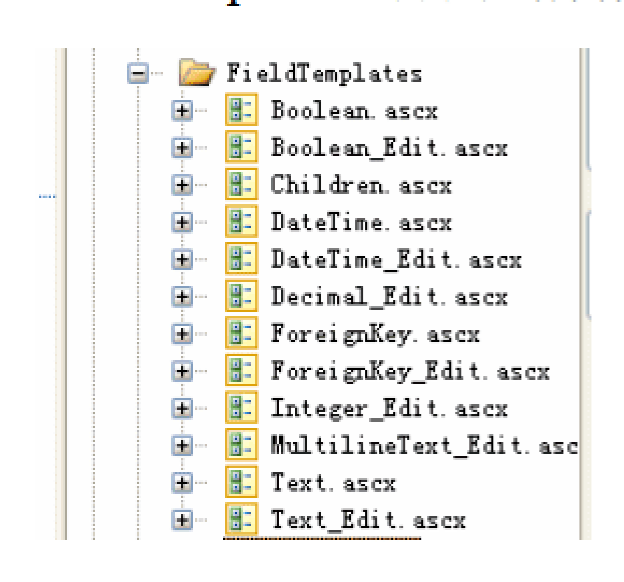


图 28.6　域模板目录

下面举例说明修改默认设置的方法。

例 28.1　对字符串格式的修改。

Text.ascx 用来定义字符串的显示格式；Text_Edit.ascx 用来定义编辑或插入字符串时的显示格式。例如打开 Text_Edit.ascx 文件，将 TextBox 控件的底色设为黄色(BackColor="Yellow")。修改后的代码如下。

```
<asp:TextBox ID="TextBox1" BackColor="Yellow" runat="server" Text='<%#
FieldValueEditString %>' CssClass="droplist"></asp:TextBox>
```

此后不论对哪张表格进行编辑时，字符串类型控件的底色都用黄色显示。

例 28.2　对时间类型数据的修改。

对时间类型数据的修改可以选用以下模板。

- DateTime.ascx. 当时间用字符串显示时，使用此模板进行编辑。
- DateTime_Edit.ascx.用于在 TextBox 控件中显示时间而且不允许为空(null)时。

因为此模板实际上包括有 RequiredFieldValidator 的校验功能。

例 28.3 对数据表视图的修改。

对网页视图的修改可在页模板 PageTemplates 中进行。此目录下是各网页中数据表的视图。默认情况下系统使用的是 List.aspx、Details.aspx、Edit.aspx、Insert.aspx 网页，并对这些网页提供了默认的界面。

打开 PageTemplates 目录时可以看见 5 种网页的视图，如图 28.7 所示。

- Details.aspx：父/子网页的界面。
- Edit.aspx：编辑时网页的界面。
- Insert.aspx：插入新记录时网页的界面。
- List.aspx：列表时网页的界面。
- ListDetails.aspx：列表及父/子综合网页的界面。

当修改了其中某种视图后，程序运行中进入该状态时即按照修改后的视图显示界面。

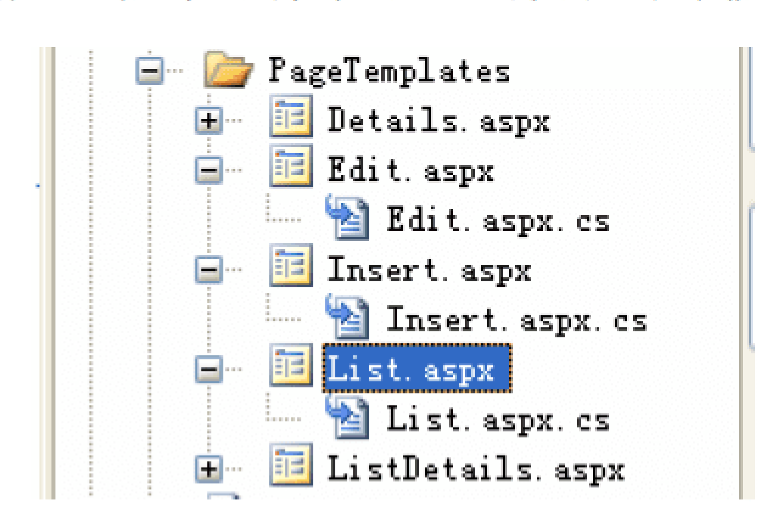


图 28.7 不同类型网页的视图

28.7 小 结

快速创建数据驱动网站是 LINQ to SQL 应用的扩展，系统提供的支架以及增强的 Global.asax 文件是快速创建的关键工具。在这些工具的支持下，能够用最快的速度创建一个具有基本功能和标准显示界面的系统，这是系统的最大特点和优点。但是，由于这个系统是用通用模板创建的，因此还缺乏个性，这是系统的一大缺点。

为此，系统还提供了方便于设计者修改的方法，就是将各类项目放在专用目录下允许进行集中修改。有了一个具有基本功能、基本界面且能够运行的系统，使用者与设计者就有了一个讨论的基础，修改和完善系统将变得更加容易，使用者与设计者的配合将更加默契。

28.8 习 题

1. 填空题

(1) 快速创建动态数据网站时，选择数据模型是在__________文件中释放或屏蔽几个字段即可。

(2) 在快速创建的数据表中所有的逻辑变量都用_________控件代替。

2. 判断题

(1) 默认情况下快速创建的数据表都具有 Ajax 功能。 ( )
(2) 支架自动提供了标准的显示界面。 ( )
(3) 支架自动提供了对数据表的操作。 ( )
(4) FieldTemplates 用来修改某个域的显示界面。 ( )

3. 简答题

(1) 支架的作用是什么?
(2) 快速创建的数据网站中 FieldTemplates(域模板)的作用是什么?
(3) 快速创建数据网站中支架起什么作用?
(4) 快速创建数据驱动网站的优点和缺点是什么?

4. 操作题

(1) 先创建一个学生成绩管理数据库，然后用快速创建数据驱动的方法创建学生成绩管理系统。

(2) 修改学生管理系统的界面。

(3) 增加学生成绩与教师情况表的连接。

第 29 章　创建电子商务网站

电子商务网站与传统的商店不同。在传统的商店中除有商品以外，还必须要有店房、门面和货架等。而电子商务网站主要依靠网络技术来完成商品的展示和交易等项工作，在电子商务网站中既没有店房、门面也没有货架，因此电子商务网站常常被称为“虚拟”商店。

电子商务网站是 Web 应用中比较广泛而又比较复杂的系统，它涉及的问题很多，但都离不开利用网络进行商品展示、客户选购、生成购货车、集中结账、生成订单等几个基本环节。本章将重点围绕这几个基本环节来讲解电子商务网站的设计。

电子商务网站的类型很多，本章选择两种比较典型的类型进行讲授。

- 食品商店网站设计。
- 服装商店网站设计。
- 账户管理。

29.1　食品商店网站设计

食品、报刊、图书等网站具有很多相似的特点。下面以食品商店为代表来讲解创建这类网站的方法。

29.1.1　概述

1. Northwind 样板库简介

SQL Server 提供的 Northwind 样板库是一个虚拟的国外食品公司的数据库，我们将利用这个数据库(加以简化)来创建自己的食品商店网站。下面先对 Northwind 样板库做一个简单的介绍。

Northwind 样板库包括十几张数据表，各表的关联以及数据表中的字段如图 29.1 所示。

Northwind 样板库的数据表包括以下 4 部分。

- 订单部分：包含订单表(Orders)、订单细目(Order Details)、运货者(Shippers)。
- 产品部分：包括类型表(Categories)、产品表(Products)、供货单位(Suppliers)。
- 雇员信息部分：包括有关雇员(Employees)的 4 张表。
- 顾客信息：包括顾客表(Customers)等。

本章将集中讲述与订单相关的部分，除此以外还需要在数据库中自行建立一个简化的订单表，订单表的结构将在下面讲述。

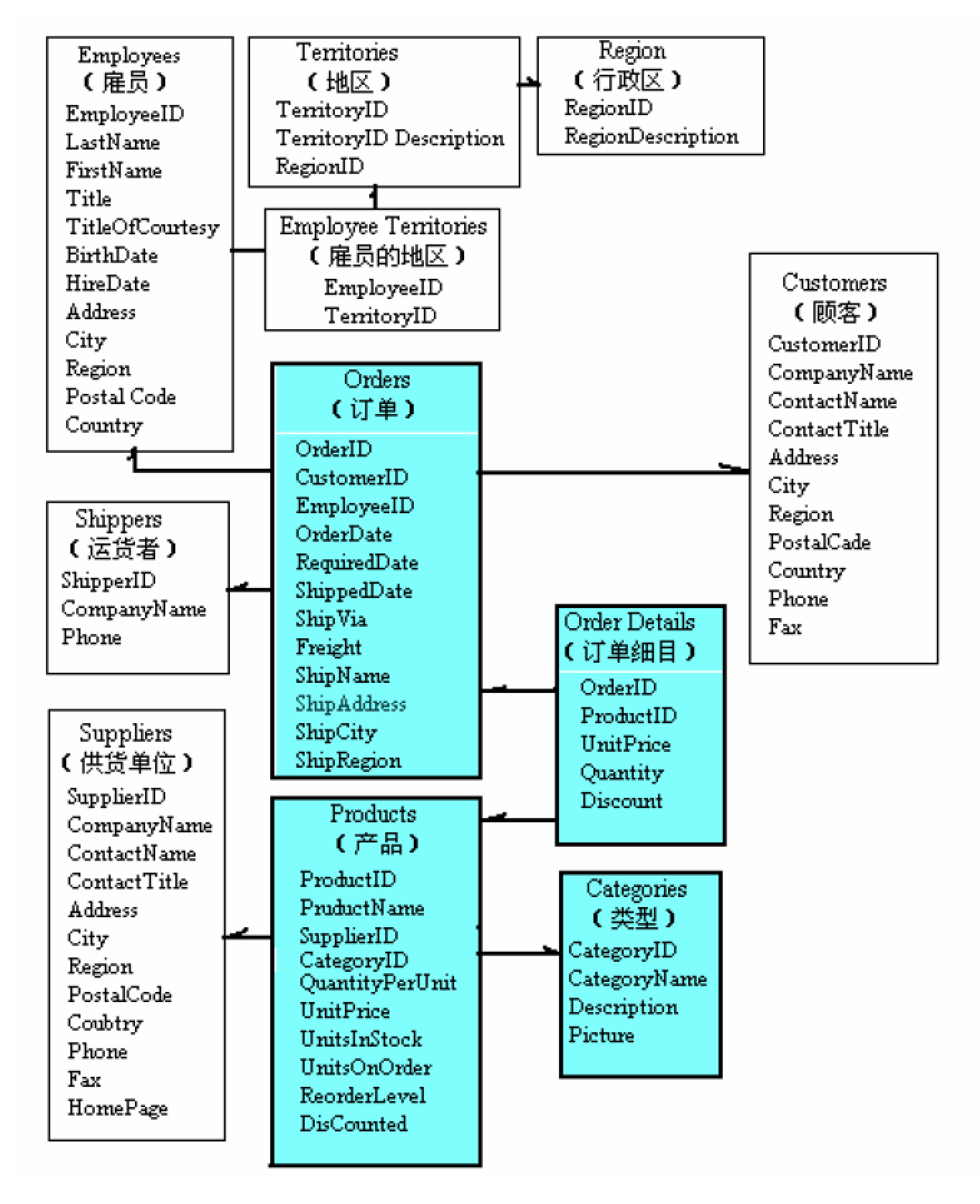


图 29.1　Northwind 样板库中数据表的结构及相互关系

2. 本系统的功能

本系统是一个食品商店的小型网站，为了使问题变得简明扼要，只讲述以下几个核心问题。

- 主页面设计。
- 商品的分类显示。
- 选购和调整购货车。
- 集中结算。
- 存储订单。
- 查看订单。

网站涉及 5 种网页，各网页的功能如图 29.2 所示。

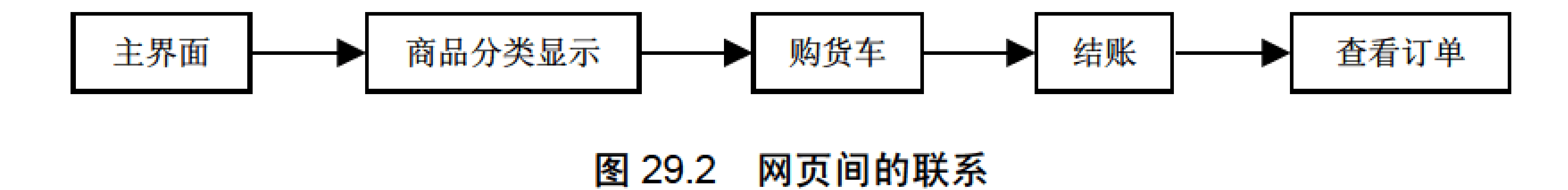


图 29.2　网页间的联系

各网页的内部结构及相互关系如图 29.3 所示。

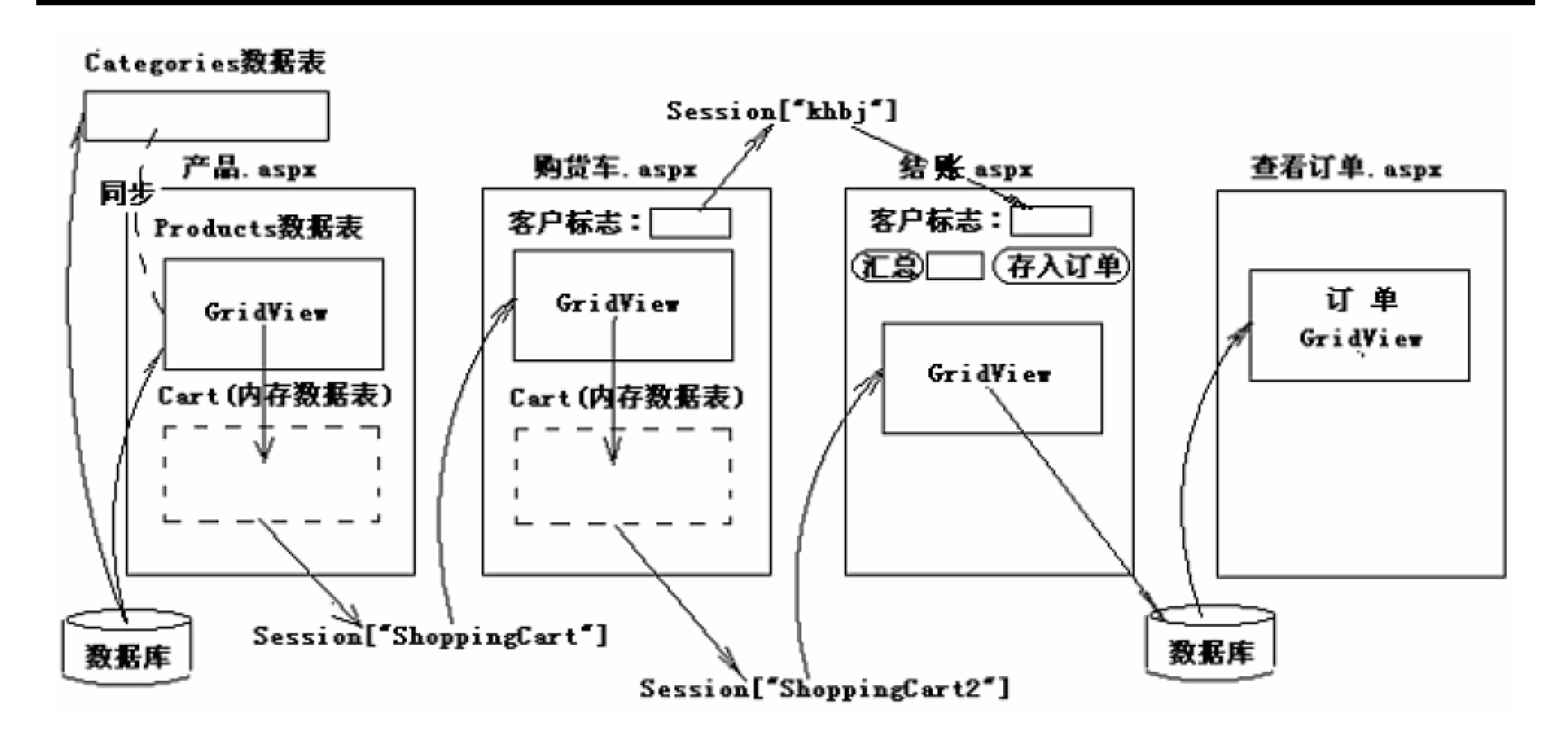


图 29.3　网页结构及信息的传递过程

理解系统的关键是搞清 GridView、内存数据表与 Session 对象三者之间的关系。系统的执行过程是：打开主界面(主界面这里没有画)，显示商品分类以及欢迎界面；在分类数据表(Categories)中任选一种数据类型，同步打开“产品.aspx”网页，用 GridView 控件分类显示 Products 数据表中相应的商品；单击想要购买的商品将该商品的数据取出来，放到下面的动态数据表中(数据表用虚线画的长方形表示)，然后再将数据表放入“购货车”中，这里的购货车用 Session["ShoppingCart"]表示；打开“购货车.aspx”网页，控件 GridView 以 Session["ShoppingCart"]作为数据源显示数据，在新网页中确定购买的数量，并输入“客户标志”；与此同时生成新的内存数据表，并将新数据表存入新的 Session["ShoppingCart2"]中；打开“结账.aspx”网页，以新 Session 对象作为数据源在 GridView 控件中显示数据，同时通过 Session["khbj"]将客户标志传入到网页中；在“结账”网页中完成计算总价的工作，最后存入订单；在“查看订单.aspx”网页中以客户标志作为查询条件显示订单，以便只显示客户本人的订单部分。

在上述各个阶段中，都允许客户对数据进行选择和修改。除开始阶段和最后阶段需要直接存取数据库以外，其他部分均采用内存动态数据表来组织数据，这样做可以提高系统的运行效率。

客户标志是识别客户的唯一标志，平时存放在客户登录表中，包括有联系地址、联系方法等。

3. 主要数据表的结构及其他准备工作

产品类型表、产品表、订单表如表 29.1、表 29.2 和表 29.3 所示。

表 29.1　产品类型表

字 段 名	类　型
CategoryID(类型标志)	int 4
CategoryName(类型名)	nvarchar 15

表 29.2　产品表

字 段 名	类　型
ProductID(产品标志)	int 4
ProductName(产品名称)	nvarchar 40
CategoryID(类型标志)	int 4
QuantityPerUnit(单位数量)	nvarchar 20
UnitPrice(单价)	money 8

表 29.3　订单表

字 段 名	类　型
客户 ID	nvarchar 20
订单 ID	int 4
产品 ID	int 4
产品名称	nvarchar 40
单位数量	nvarchar 20
单价	numeric
订购量	int 4
订购时间	datetime 8

其中产品类型表和产品表分别直接采用 SQL Server 中的样板库 Northwind 中的 Categories 表和 Products 表。简化的订单表则需要自行创建。

4. 其他准备工作

为了便于网站管理，先在网站中设置几个子目录，分配客户的角色，确定访问策略。具体做法见第 21 章。

29.1.2　主界面设计

为了使得系统具有很好的可重用性和可维护性，应该尽可能使用用户控件(User Control)和母版页(Master Page)以便发挥代码重用的优势以减少重复劳动，并且使得各个网页的显示风格一致。

1. 创建用户控件

商店的商标、网页之间链接的图标以及查询部分是大多数网页都需要显示的部分，可以先将其建成用户控件。例如这里将商店商标、几个网站浏览的控件(hyperLink)以及查询界面等分别做成用户控件。用户控件中的代码可以以后再补充上去。

2. 创建母版页

先创建母版页然后再创建模板中的网页比较方便。母版页的布局如图 29.4 所示。



图 29.4 母版页的布局

母版页的上方放的是用户控件，左下方放的是 GridView 控件，该控件与类型表(Categories)进行数据绑定，右下方是给各网页留下的空间。

3. 设计主页

主页是客户访问的第一个页面，可以给客户提供第一印象。它的主要任务是吸引客户并引导客户打开选购界面。一个好的主页应该生动、清晰，能够激发客户购买的兴趣。

在母版页中生成一个主页，设计欢迎界面，其简要情况如图 29.5 所示。



图 29.5 欢迎界面

在母版页中分别再生成其他网页。例如“商品.aspx”、“购货车.aspx”、“结账.aspx”、“订单.aspx”等。这些网页的内容可以以后再补充。在用户控件中将链接指针分别与各个网页链接。

4. 产品类型与产品目录之间同步

为了使得产品类型表与产品表之间取得同步，在显示类型的 GridView 的 Column 属性中增添一“超链接列”，并且为此列设置同步所需的 URL、“URL 字段”、“URL 格式字符串”等，使得单击该超链接按钮时向子表传送出同步字段 CategoryID。具体设置情况如图 29.6 所示。

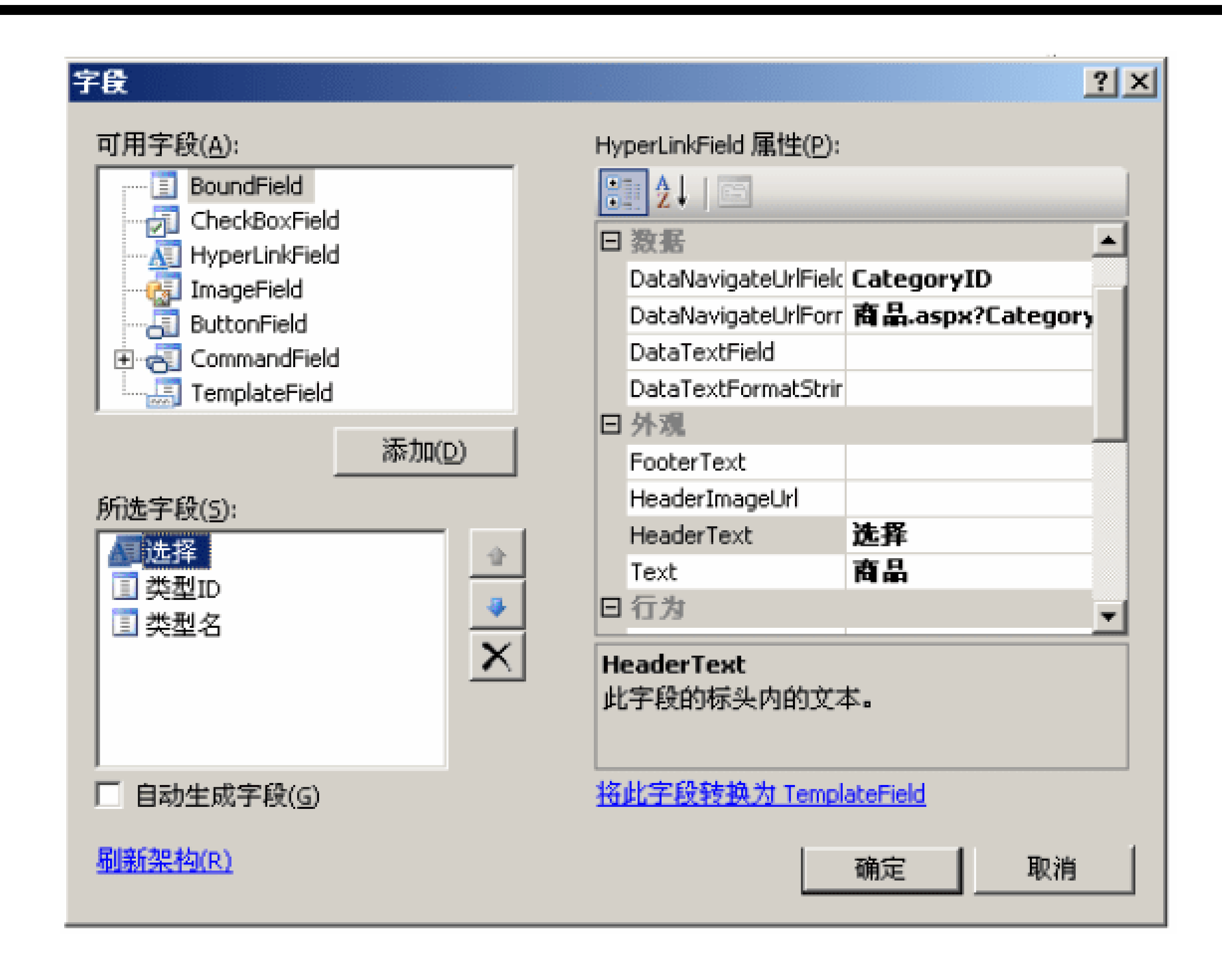


图 29.6　在类型表中设置同步字段

在产品表的网页中利用 QueryString 设置查询语句，以便根据属性表传来的参数进行查询，以达到两表同步的目的。在产品表中设置查询语句时的设置如图 29.7 所示。

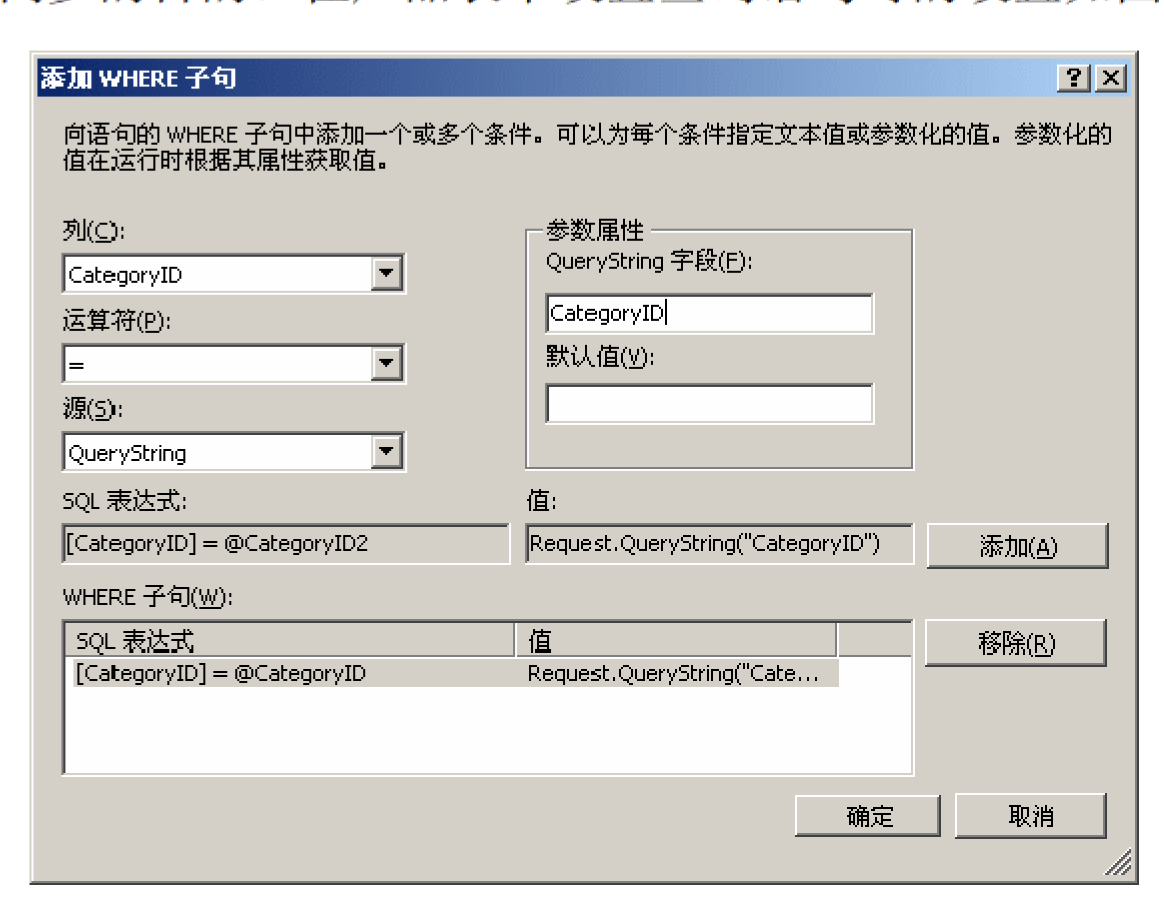


图 29.7　在产品表中建立同步关系

两表同步的结果如图 29.8 所示。



选择	类型ID	类型名
商品	2	调味品
商品	5	谷物类
商品	8	海鲜
商品	7	农产品
商品	6	肉，家禽
商品	4	乳汁品
商品	3	糖果，蜜饯
商品	1	饮料

选择	产品ID	产品名	类型ID	单元数量	单价
选购	8	蟹	8	每箱24瓶	19.0000
选购	9	龙虾	8	每箱24瓶	48.0000
选购	17	虾米	8	每箱24瓶	19.0000
选购	19	海参	8	每箱24瓶	19.0000
选购	28	鱿鱼	8	每箱24瓶	19.0000
选购	29	干贝	8	每箱24瓶	19.0000

图 29.8　两表之间的同步显示

29.1.3 选择商品

为了选择商品，在产品表中的栏目(Column)中增加一按钮(Button)，并且在它的CommdName 属性中取名为 select(也可以取其他名字)。当单击该按钮时，将该产品的副本取出来动态生成数据表，利用这个动态数据表来生成购货车。

为了生成这个数据表，需要解决以下三个问题。

- 单击按钮时将触发什么事件。
- 如何提取 GridView 控件中的数据。
- 如何创建动态数据表并将取出的数据放入其中。

1. 选择按钮触发的事件

GridView 控件中的 RowCommand 事件，是 GridView 控件内任一按钮都将触发的事件。因此在使用这个事件时，先要判断当前单击的是哪个按钮。判断的方法是根据按钮的命令名，语句如下。

```
if (e.CommandName=="按钮命令名")
  {…}
```

2. 取出相关的数据

根据单击的行以及在 GridView 中的列，可以取出相应的数据。确定单击行的语句如下。

```
int index = Convert.ToInt32(e.CommandArgument);
GridViewRow row = GridView1.Rows[index];
```

取出某列的语句如下。

```
        string bhText = row.Cells[1].Text;
        string mcText = row.Cells[2].Text;
```

上述语句提取出单击的行中第二列和第三列中的数据(列的序号从0开始)。

29.1.4 创建购货车

1. 动态生成内存数据表

下面将用 Session["ShoppingCart"]代表购货车。为了创建购货车，先动态生成内存数据表，然后将数据表放入购货车中。

为此先定义数据表的结构，即定义数据表各列的数据类型及标题。语句如下。

```
System.Data.DataTable Cart = new System.Data.DataTable();//定义数据表对象
        if (e.CommandName == "select")
        {
            if (Session["ShoppingCart"] == null)
            {
                Cart.Columns.Add("商品编号", typeof(int));
                                         //确定各列的标题及类型
                Cart.Columns.Add("商品名称", typeof(string));
```

```
                Cart.Columns.Add("单元含量", typeof(string));
                Cart.Columns.Add("单价", typeof(double));
                Session["ShoppingCart"] = Cart;
            }
        //再取出各列的数据，同时进行类型转换
        Cart = (System.Data.DataTable)Session["ShoppingCart"];
            int index = Convert.ToInt32(e.CommandArgument);
            GridViewRow row = GridView1.Rows[index];
            string bhText = row.Cells[1].Text;
            string mcText = row.Cells[2].Text;
            string dyText = row.Cells[4].Text;
            string djText = row.Cells[5].Text;
            int bh = int.Parse(bhText);
            double dj = double.Parse(djText);
            //然后增加数据行，并将数据填入该行的各列之中
            System.Data.DataRow rr = Cart.NewRow();
            rr["商品编号"]=bh;
            rr["商品名称"] = mcText;
            rr["单元含量"] = dyText;
            rr["单价"] = dj;
            Cart.Rows.Add(rr);  // 将新行加入数据表中
            Session["ShoppingCart"] = Cart;  …
        }
```

由于在默认情况下，网页之间不保持状态，当选择多行时，购货车中只会保留最后的选择。为了能够在购货车中保留多项选择，需要利用 Session 对象来保持状态。语句如下。

```
if(Session["ShoppingCart"]==null)
     {
       Cart = new DataTable();
        …
       Session["ShoppingCart"]=Cart;
     }
     Cart = (System.Data.DataTable)Session["ShoppingCart"];
```

综上所述，创建内存数据表的完整代码如下。

```
<script runat="server">
    void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        System.Data.DataTable Cart = new System.Data.DataTable();
         //生成内存数据表对象
        if (e.CommandName == "select")
        {
            if (Session["ShoppingCart"] == null)  // 定义数据表结构
            {
              Cart.Columns.Add("商品编号", typeof(int));
              Cart.Columns.Add("商品名称", typeof(string));
              Cart.Columns.Add("单元含量", typeof(string));
              Cart.Columns.Add("单价", typeof(double));
              Session["ShoppingCart"] = Cart;
```

```
            }
            Cart = (System.Data.DataTable)Session["ShoppingCart"];
            int index=Convert.ToInt32(e.CommandArgument);    //确定单击的行
            GridViewRow row = GridView1.Rows[index];
            string bhText = row.Cells[1].Text;               //取出列中的数据
            string mcText = row.Cells[2].Text;
            string dyText = row.Cells[4].Text;
            string djText = row.Cells[5].Text;
            int bh = int.Parse(bhText);                      //数据的类型转换
            double dj = double.Parse(djText);
            System.Data.DataRow rr = Cart.NewRow();
            rr["商品编号"]=bh;                 //将取出的数据放入数据表中
            rr["商品名称"] = mcText;
            rr["单元含量"] = dyText;
            rr["单价"] = dj;
            Cart.Rows.Add(rr);                //将新行加入数据表中
            Session["ShoppingCart"] = Cart;   // 将数据表放入 Session 中
        }
        }
</script>
```

2. 显示购货车的网页设计

在显示购货车的网页中将 Session["ShoppingCart"]作为数据源显示在 GridView 中，语句如下。

```
private void Page_Load(object sender, System.EventArgs e)
{
  GridView1.DataSource= Session["ShoppingCart"];
  DataBind();
}
```

购货车窗体的界面如图 29.9 所示。



图 29.9 购货车的窗体界面

在本窗体中增加了以下设置。

(1) 增加了客户标记的输入窗口(TextBox 控件)。客户标记必须唯一、可靠，根据该

标记能够与客户联系，确定发货的方法和地址，为此要增加 RequiredFieldValidator 校验控件，以防止输入中遗漏。

(2) 在 GridView 的【编辑列】中增加一个按钮，并为它的 CommandName 属性命名(例如命名为 buy)，以便确认选择项。

(3) 在 GridView 的【编辑列】中增加一个模板列，将 TextBox 控件放入 ItemTemplate 模板中，以便输入购买数量。先将其默认值设为 1。

(4) 在窗体页的 Page_Load 事件中加上判断条件：if(!IsPostBack) { … }。

IsPostBack 属性是用来区分网页是第一次被打开，还是后续打开(事件处理后的返回)。当网页为后续打开时，该属性为 true，否则为 false。

```
if(!IsPostBack)
{
  // 执行语句
}
```

代码表明只有网页第一次(不是后续)打开时才执行大括号中的语句。

在本网页中应写入以下代码。

```
private void Page_Load(object sender, System.EventArgs e)
{
  if(!IsPostBack)
  {
    GridView1.DataSource= Session["ShoppingCart"];
    DataBind();
  }
}
```

这表明只有第一次打开网页时，GridView1 的数据源来自 Session[…]，后续连接时不再受 Session[…]的限制，因而允许修改购买的数量。

(5) 数据表中增添了几个字段。如订购数量、合计等(见后面的代码)，以便与订单表一致，因此需要为新数据表使用另一个 Session 对象(这里使用 ShoppingCart2)。

在 GridView 的 RowCommand 事件中编写如下代码。

```
void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
 {
  System.Data.DataTable Cart = new System.Data.DataTable();//生成数据表
  if (e.CommandName == "buy")
   {
    if (Session["ShoppingCart2"] == null)
     {
      Cart.Columns.Add("商品编号", typeof(int));              //建立数据表结构
      Cart.Columns.Add("商品名称", typeof(string));
      Cart.Columns.Add("单元含量", typeof(string));
      Cart.Columns.Add("单价", typeof(double));
      Cart.Columns.Add("订购数量", typeof(int));
      Cart.Columns.Add("折扣", typeof(double));
      Cart.Columns.Add("合计", typeof(double));
      Session["ShoppingCart2"] = Cart;
     }
```

```
    Cart = (System.Data.DataTable)Session["ShoppingCart2"];
   if (TextBox2.Text == "")      //输入客户标志
    {
    Validate();                  //调用校验控件进行校验
    }
   else
   {
    Session["khbj"] = TextBox2.Text;
    int index = Convert.ToInt32(e.CommandArgument);
    GridViewRow row = GridView1.Rows[index];
    TextBox tt = (System.Web.UI.WebControls.TextBox)row.Cells[1].FindControl
("TextBox1");
    string dgl = tt.Text;
    int dg = int.Parse(dgl);
    if (dg <= 1) dg = 1;         //若购货量小于 1 时，定为 1
    string bhText = row.Cells[2].Text;
    string mcText = row.Cells[3].Text;
    string dyText = row.Cells[4].Text;
    string djText = row.Cells[5].Text;
    int bh = int.Parse(bhText);
    double dj = double.Parse(djText);

    System.Data.DataRow rr = Cart.NewRow();
    rr["商品编号"]=bh;
    rr["商品名称"] = mcText;
    rr["单元含量"] = dyText;
    rr["单价"] = dj;
    rr["订购数量"] = dg;
    int zk = 1;
    rr["折扣"] = zk;
    double hj = dj * dg * zk;    //合计使用三者的乘积
    rr["合计"] = hj;
    Cart.Rows.Add(rr);
    Session["ShoppingCart2"] = Cart;
   }
  }
 }
```

29.1.5 结账

1. 结账网页的数据显示

在结账网页中的 GridView 控件应该以 Session["ShoppingCart2"]作为数据源来显示数据，除此而外还应该将客户标记显示出来，并且使用 if(!isPostBack){…}判断语句，以便客户对某些可选的参数进行选择。其语句如下。

```
GridView1.DataSource = Session["ShoppingCart2"];
```

2. 在 GridView 控件中增加复选框

为了允许客户在结账时有机会对自己的选择做进一步调整，可以在结账界面的每条记

录前面增加一复选框。默认情况下这些复选框都被选中，如果想撤销该项选择时，只须取消该复选框的选择即可，订单表中将不会存入被取消的选项。

为了增加复选框，需要在 GridView 中通过 Column 属性增添一“模板”字段，并且在模板中增添复选框控件。模板的设置代码如下。

```
<Columns>
  <asp:TemplateColumn HeaderText="保存">
    <ItemTemplate>
      <center>
        <asp:CheckBox ID="Check1" Checked="True" Runat="server" />
      </center>
    </ItemTemplate>
  </asp:TemplateColumn>
</Columns>
```

结账的界面如图 29.10 所示。

网络食品商店　主页　购货车　结账　订单　查询：

选择	类型ID	类型名
商品	2	调味品
商品	5	谷物类
商品	8	海鲜
商品	7	农产品
商品	6	肉，家禽
商品	4	乳汁品
商品	3	糖果，蜜饯
商品	1	饮料

客户标志：cheng　汇总　938　存入订单

确认	商品编号	商品名称	单元含量	单价	订购数量	折扣	合计
☑	9	龙虾	每箱24瓶	48	9	1	432
☑	19	海参	每箱24瓶	19	19	1	361
☑	5	酱油	每箱24瓶	29	5	1	145

图 29.10　结账的界面

在这个界面中，客户标志和表格中的数据均从上一个网页传来。在结账界面主要完成汇总计算以及将订单表存入数据库的操作。

3. 汇总的计算

当单击【汇总】按钮时，文本框中将显示总和。如果改变了复选框的选择，汇总的结果也应该跟着改变。为了进行汇总计算，需设置循环语句，逐条检查复选框的选择状态，只有该复选框被选中时，才将该记录的数据计入汇总中。此处需要用到的类和方法如下。

- GridView1.Rows[ii]：用来表示 GridView 控件中的某一行。
- FindControl("控件的 id")方法：用来在当前的命名容器中搜索带指定 id 参数的服务器控件。

【汇总】按钮的代码如下。

```
void Button2_Click(object sender, EventArgs e)
 {
   double sum=0.0;
   for(int ii=0; ii < GridView1.Rows.Count;ii++)
   {
```

```
    CheckBox                                cc                                =
(CheckBox)GridView1.Rows[ii].Cells[0].FindControl("CheckBox1");
    if(cc.Checked)                    //若复选框被选中
      {
        sum = sum + (double.Parse(GridView1.Rows[ii].Cells[7].Text));
      }
   }
   TextBox2.Text=sum.ToString();     //显示汇总结果
 }
```

为实现复选框的功能，在 Page_Load 事件中需要做一些改变。代码如下。

```
void Page_Load(object sender, EventArgs e)
 {
   if (!IsPostBack)                  //为了允许对复选框的修改
    {
     TextBox1.Text = (string)Session["khbj"];           //显示客户标记
     GridView1.DataSource = Session["ShoppingCart2"];   //连接数据源
     GridView1.DataBind();
    }
}
```

29.1.6 保存及显示订单

1. 保存订单

如果客户对结果感到满意时，可单击【存入订单】按钮，将订单存入数据库的订单表中。存入时使用存储过程，为此需要先建立订单表，表的结构如前面所示，另外还需要在数据库端创建存储过程，这里需要创建一个增加新记录的存储过程。使用的语句如下。

```
INSERT INTO dbo.订单表(客户 ID, 产品 ID, 产品名称, 单位数量, 单价, 订购量, 订购时间) VALUES (@客户 ID, @产品 ID, @产品名称, @单位数量, @单价, @订购量, @订购时间);
```

在数据库端设置的存储过程如图 29.11 所示。



图 29.11 存入订单的存储过程

为了调用存储过程，从工具箱中拖入一个 SqlDataSource 控件并与存储过程连接。连接过程中最大的不同点是对待定参数赋值的方式。这些待定参数的值不是从固定的控件中读取，而是通过循环语句从 GridView 控件(代表内存中的订单)的字段中获得。

待定参数的赋值过程是：开始对 GridView 控件逐条记录进行循环；在循环语句中首先取出复选框的状态，判断复选框是否被选中；如果被选中，先清除原有参数，然后给各参数赋值；最后调用存储过程以存入订单。具体的代码如下。

```
void Button1_Click(object sender, EventArgs e)
 {
for (int ii=0; ii<GridView1.Rows.Count; ii++)      //循环语句
{
   //先判断复选框是否被选择
CheckBox
cc=(CheckBox)GridView1.Rows[ii].Cells[0].FindControl("CheckBox1");
  if (cc.Checked)        //如果被选中则取出数据存入数据表中
{
SqlDataSource1.InsertParameters.Clear();
SqlDataSource1.InsertParameters.Add("客户 ID",Session["Khbj"].ToString());
SqlDataSource1.InsertParameters.Add("产品 ID",
                GridView1.Rows[ii].Cells[1].Text);
SqlDataSource1.InsertParameters.Add("产品名称", GridView1.Rows[ii].
                Cells[2].Text);
SqlDataSource1.InsertParameters.Add("单位数量", GridView1.Rows[ii].
                Cells[3].Text);
SqlDataSource1.InsertParameters.Add("单价", GridView1.Rows[ii].
                Cells[4].Text);
SqlDataSource1.InsertParameters.Add("订购量",
                GridView1.Rows[ii].Cells[5].Text);
SqlDataSource1.InsertParameters.Add("订购时间", DateTime.Today.
                ToShortDateString());
SqlDataSource1.InsertCommand = "cccc.NewCartInsert";
Session["ccsj"] = DateTime. Today.ToShortDateString();
SqlDataSource1.Insert();
}
}
}
```

其中“Session["ccsj"] = DateTime. Today.ToShortDateString()”的作用是将存储时间记入 Session 中，以便查看订单。NewCartInsert 为存储过程名，cccc 为存储过程的所有者。默认情况下所有者为 dbo。

2. 查看订单

为了保证客户只能查看自己的订单，订单的显示应该以客户标志(Session["Khbj"])以及当天的日期(Session["ccsj"])作为查询条件。为此在“查看订单.aspx”网页中设置 GridView 控件，并通过数据源控件利用 Session 设置查询条件(“客户 ID”与“订购时间”)。设置的方法如图 29.12 所示。

利用上述语句可以查看客户当天的订单。除此而外，为了给客户最后的修改机会，还可以在显示订单表中增加【删除】按钮，允许客户删除自己当天的订单。具体的设置方法此处不再赘述。

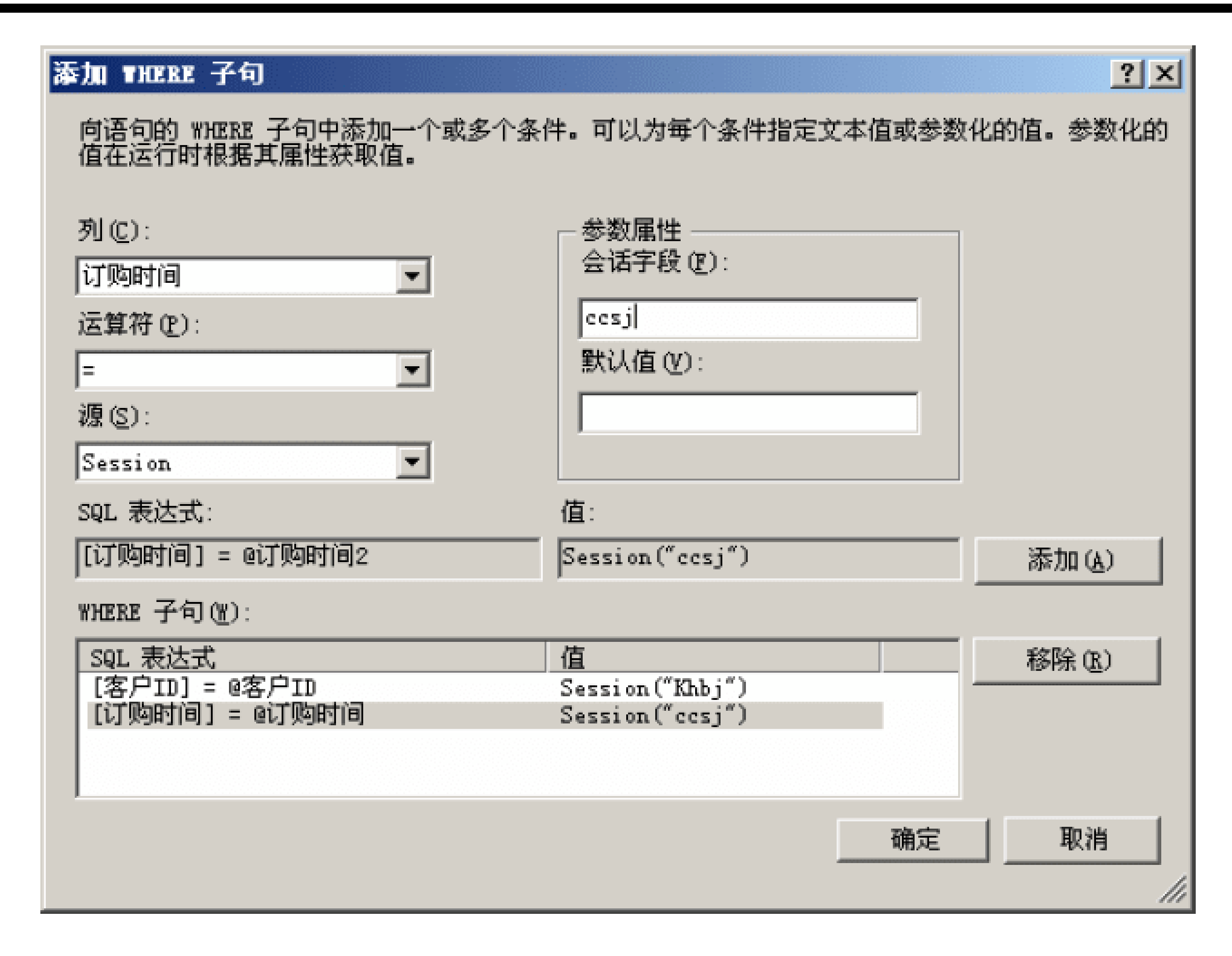


图 29.12　查看订单时设置查询条件

最后显示订单的界面如图 29.13 所示。

客户标志: 刘丽1385　　返回结帐台　　返回首页

删除	客户ID	订单ID	产品ID	产品名称	单位数量	单价	订购量	订购时间
删除	刘丽1385	66	4	麻油	每箱12瓶	22.00	5	2004-5-4 0:00:00
删除	刘丽1385	67	10	味精	每包2公斤	31.00	6	2004-5-4 0:00:00
删除	刘丽1385	68	19	海参	每包2公斤	9.20	8	2004-5-4 0:00:00
删除	刘丽1385	69	6	胡椒粉	每包2公斤	25.00	7	2004-5-4 0:00:00

图 29.13　显示订单示例

29.1.7　放大图像介绍商品的方法

在前面的商品界面中，是用表格来显示商品的。采用这种形式的好处是可以将多条记录集中在一起，便于客户进行选择、比较。对于食品等类型的商品来说，采用这种形式比较合适。但是对于服装、首饰、花卉等类型的商品来说，这种形式就存在着较大的缺陷。因为顾客选择这类商品时，外观常常是选择的主要依据。虽然在表格中也能插入商品的图像，但由于受表格的限制，图像往往比较小，不能满足需要。

下面介绍用放大图像并结合少量文字来介绍商品的方法。

为此增加一个用于图像显示的网页(商品 2.aspx)，该网页与原来的表格形式的网页(商品.aspx)同步，ProductID 是两表的同步字段。在显示图像的网页中仍然采用 GridView 控件，因为利用 GridView 控件时，数据绑定和数据访问都非常方便。但另一方面将利用控件的模板来摆脱表格的限制，以充分发挥模板中可以任意布局的特点，来构建需要的图像界面。具体设计的要点如下。

(1)　在 Product 数据表中增加两个字段：imagePath 和 description。前面的字段用来存入图像在网站中的路径，后面的字段用来存入对商品的简要描述。

(2) 在原来 “商品.aspx” 网页的 GridView 控件中增加一个 HyperLink 字段，以便与图像网页(商品 2.aspx)进行同步，同步的字段是 ProductID。新字段属性的设置如图 29.14 所示。

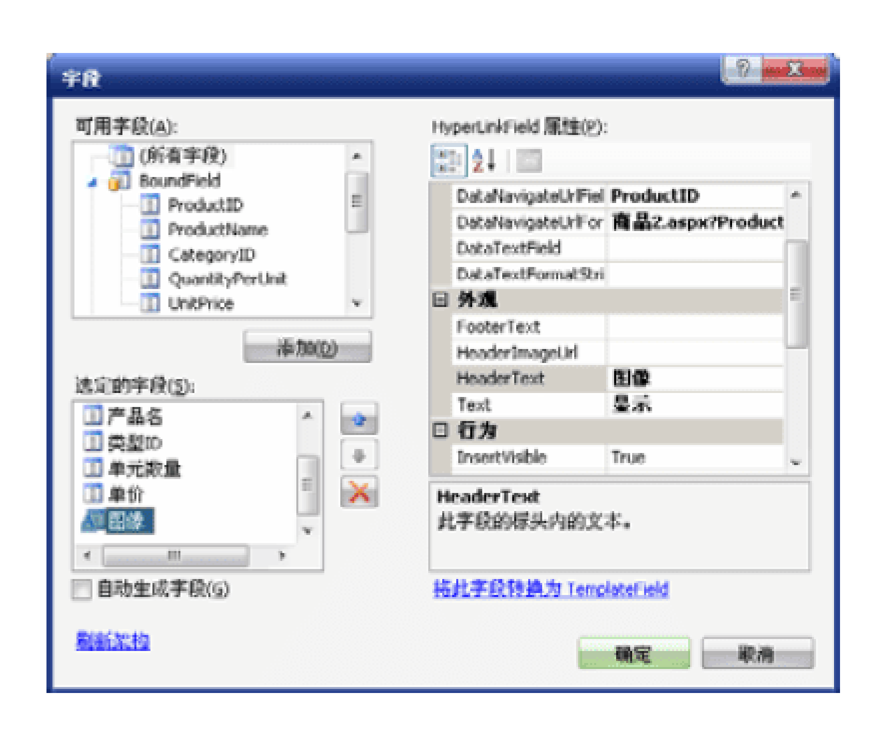


图 29.14　在商品的表格中增加同步字段

注意将同步字段的 Target 属性设为 “_blank” (这一点图上没有显示)，即同步时将新网页放在一个打开的空窗体中。

(3) 打开新网页(商品 2.aspx)，并在新网页中放入 GridView 1 控件，通过数据源控件(SqlDataSource1)与 Product 数据表相连。作为子表，通过 ProductID 字段利用 QueryString()方法与主表取得同步。同步方法如 12.2.3 节所述，这里不再重复。

设置后的语句是

```
SELECT [ProductID], [ProductName], [CategoryID], [QuantityPerUnit],
[UnitPrice], [imagePath], [discription]
    FROM [Products]
    WHERE ([ProductID] = @ProductID)
```

(4) 增加模板字段。打开 “商品 2.aspx” 网页中 Gridview1 的【编辑列】窗口。清除所有被选定的字段后，再放进一个模板字段(TemplateField)来取代它们。

(5) 编辑模板。右击 Gridview1，选择【编辑模板】命令，选择 Column[0]对模板进行编辑，如图 29.15 所示。

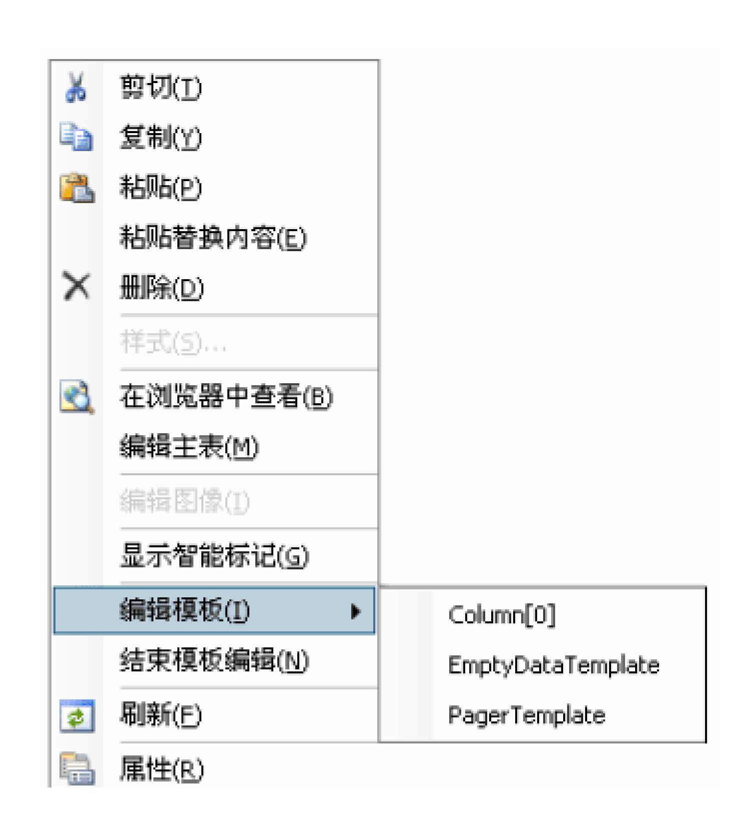


图 29.15　选择需编辑的模板

① 对模板进行布局。布局的情况如图 29.16 所示。

在图 29.16 所示的布局中，左上方放入一个 Image1 控件来显示商品的图像，右边标上“名称”和“单价”两个字段，下面标上“简单描述”，最下面放入一个 HTML 的 Button 控件(用于返回原界面)。



图 29.16 在模板中的布局

② 进行数据绑定。

下面用 Eval()或 Bind()方法来绑定数据。模板中的语句如下。

```
<asp:GridView ID="GridView1" runat="server"
DataSourceID="SqlDataSource1" AutoGenerateColumns="False" Height="274px"
Width="478px">
   <Columns>
      <asp:TemplateField>
         <ItemTemplate>
           <table style="width: 100%; height: 100%; border-right: black
thin solid; border-top: black thin solid; border-left: black thin solid;
border-bottom: black thin solid; background-color: #ffffff; direction:
ltr; text-indent: 8pt;">
             <tr>
                <td rowspan="4" style="width: 18px">
                  <asp:Image ID="Image1" ImageUrl=<%# Eval("imagePath") %>
runat="server" BorderColor="#0000C0" BorderStyle="Solid"
BorderWidth="1px" ImageAlign="Left" />
               </td>
               <td style="width: 100px">
               </td>
             </tr>
             <tr>
                 <td style="width:640px">
                   <h4 align="left">名称: <%# Eval("ProductName") %></h4>
                   </td>
             </tr>
             <tr>
                 <td style="width: 640px">
                 <h4 align="left"> 数量: </h4><h5> <%#
```

```
Eval("QuantityPerUnit") %></h5></td>
              </tr>
              <tr>
                 <td style="width: 640px; height: 21px">
                    <h4 align="left"> 单价: </h4><h5><%# Eval("UnitPrice")
                    %></h5>
                    </td>
              </tr>
              <tr>
                 <td colspan="2" style="height: 59px">
                    <h4> 简单描述:</h4><h5><%# Eval("discription")
                    %></h5></td>
               </tr>
               <tr>
                  <td style="width: 18px" >
                    <input id="Button1" type="button" style="background-
                    color:honeydew; width: 62px;"  value="返回..."
                    onclick="javascript:window.close()" />
                  </td>
               </tr>
            </table>
          </ItemTemplate>
          <HeaderTemplate>
                商品简介
          </HeaderTemplate>
        </asp:TemplateField>
     </Columns>
   <HeaderStyle BackColor="#E0E0E0" />
</asp:GridView>
```

现在的每个模板代表一条记录，代码中：

- <%# Eval("ProductName") %>代表显示商品名。
- <%# Eval("QuantityPerUnit") %>代表显示单元数量。
- <%# Eval("UnitPrice") %>代表显示单价。
- <%# Eval("discription") %>代表显示简要介绍。
- <asp:Image ID="Image1" ImageUrl=<%# Eval("imagePath") %> runat="server" />代表将图片的路径赋给 Image 的属性 ImageUrl。

(6) 设置【返回】按钮的代码。此按钮是一个浏览器控件。当被单击时，使用 JavaScript 语言调用 window.close()方法关闭同步时打开的窗口。代码如下。

```
<input id="Button1" type="button" style="background-color:Lime" value="返回..."
    onclick="javascript:window.close()" />
```

(7) 运行程序。单击商品表格中图像字段的【显示】按钮，打开同步的图形界面，如图 29.17 所示。



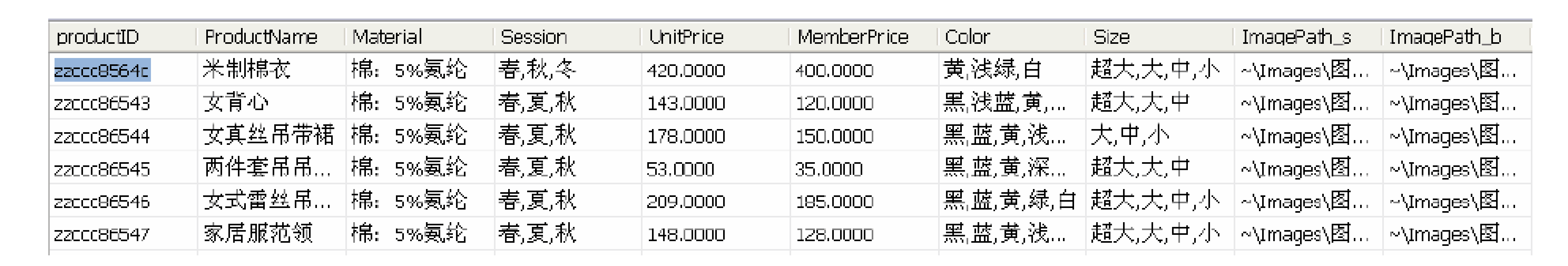
图 29.17　表格界面与图像界面同步

单击图形界面中的【返回】按钮时，将关闭图形显示窗口，回到表格的界面。如果确定购买时，单击表格窗口中的【选购】按钮，生成购货车及订单的操作如前面所述。

29.2　服装商店网站设计

服装商店与首饰、花卉、手机等网站有很多相似的特点，它们对显示界面的要求都比较高，购买时可选的参数也比较多。

现在假定某服装商品的数据简表如图 29.18 所示。



productID	ProductName	Material	Session	UnitPrice	MemberPrice	Color	Size	ImagePath_s	ImagePath_b
zzccc8564c	米制棉衣	棉：5%氨纶	春,秋,冬	420.0000	400.0000	黄,浅绿,白	超大,大,中,小	~\Images\图...	~\Images\图...
zzccc86543	女背心	棉：5%氨纶	春,夏,秋	143.0000	120.0000	黑,浅蓝,黄,...	超大,大,中	~\Images\图...	~\Images\图...
zzccc86544	女真丝吊带裙	棉：5%氨纶	春,夏,秋	178.0000	150.0000	黑,蓝,黄,浅...	大,中,小	~\Images\图...	~\Images\图...
zzccc86545	两件套吊吊...	棉：5%氨纶	春,夏,秋	53.0000	35.0000	黑,蓝,黄,深...	超大,大,中	~\Images\图...	~\Images\图...
zzccc86546	女式雷丝吊...	棉：5%氨纶	春,夏,秋	209.0000	185.0000	黑,蓝,黄,绿,白	超大,大,中,小	~\Images\图...	~\Images\图...
zzccc86547	家居服范领	棉：5%氨纶	春,夏,秋	148.0000	128.0000	黑,蓝,黄,浅...	超大,大,中,小	~\Images\图...	~\Images\图...

图 29.18　服装商品数据表

表中 ProductName 为商品名；Material 为材质；Session 为季节；MemberPrice 为会员价；Color 为颜色；Size 为尺寸；ImagePath-s 为小图路径；ImagePath-b 为大图路径。

其中可选的参数包括大小、颜色、材质等。

29.2.1　几张网页之间的联系

几张网页之间的联系如图 29.19 所示。

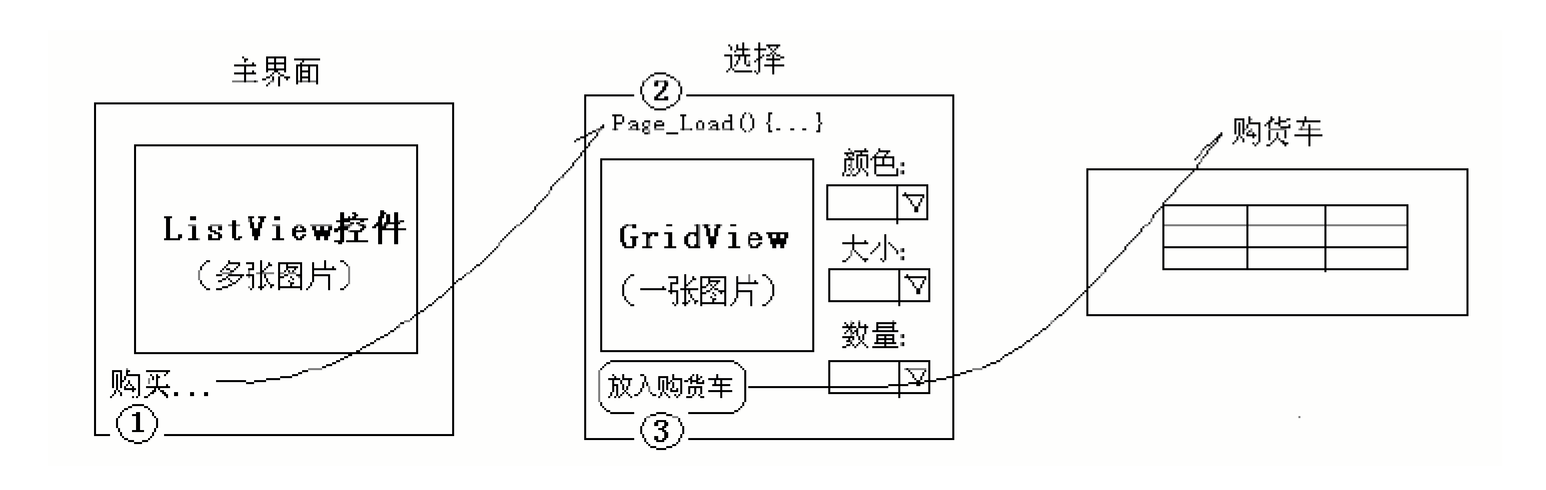


图 29.19　几张网页的联系

图中：

① 代表单击【购买...】按钮时转向新网页，执行跨页同步功能。

② 代表生成网页时，动态生成几个下拉列表控件。

③ 代表单击【放入购货车】按钮时，生成购货车。

以上三个阶段的代码将在下面讲述。

29.2.2　主界面的设计

主界面采用 ListView 控件，其设计方法参见第 15 章。在配置 ListView 中，选择【平铺】布局，并在 ItemTemplate 模板中进行布局和数据绑定，如图 29.20 所示。



图 29.20　选择 ItemTemplate 模板

确定每行包括几条记录(3 条记录，3 列)以及每页的大小(6 条记录)后，程序运行的界面如图 29.21 所示。



图 29.21　服装商店主界面

注意： 界面中对单价的显示使用了删除的标签，即用<s>...</s>来表示对会员不使用单价而使用会员价。

29.2.3 跨页同步

当单击【购买...】按钮时将转向新的网页，并同时将 productID 的值传送过去。代码如下。

```
<a href="选择.aspx?id=<%# Eval("productID") %>">购买...</a>
```

在新网页(选择.aspx)中，放进 GridView 控件，并进行下列的设置。

(1) 通过数据源控件连接到服装数据表，并且通过 QueryString 方法与传来的 id 同步。

(2) 在【编辑列】中去掉所有生成的字段，然后增添 TemplateField 模板，并且在模板中按照图 29.22 的方式给控件进行布局。



图 29.22 选择界面

29.2.4 动态生成控件

为了动态生成下拉列表，首先需要获得数据表中的相关数据，为此利用两个 Label 控件作为中间环节来传递数据，让它们分别与数据表中的 Color 和 Size 字段进行数据绑定，但是将它们的 Visible 属性设为 false(即不显示这两个控件)。其代码如下。

```
<asp:Label ID="color1" runat="server" Text='<%# Eval("Color") %>'
Visible="false" />
<asp:Label ID="Size1" runat="server" Text='<%# Eval("Size") %>'
Visible="false" />
```

然后在网页中拖入两个 DropDownList 控件，将前面两个 Label 中绑定的字符串进行分解(这里是依据逗号进行分解)，并将分解后的字段分别放入两个 DropDownList 控件中。其代码如下。

先在类中定义一个成员。

```
GridViewRow row;//因为在下面两个方法中都要使用这个成员
protected void Page_Load(object sender, EventArgs e)
 {
```

```
    row = GridView1.Rows[0];
    if (!IsPostBack)
     {
       Label lc = (Label)row.FindControl("color1");
       string cc = lc.Text;            //取出数据表中的颜色选项
       string[] fgcc = cc.Split(','); //以逗号为分隔符，将字符串分解为字符串数组
       int count1 = fgcc.GetLength(0);//确定数组的个数
       DropDownList dw1 = (DropDownList)row.FindControl("DropDownList1");
                  //取出颜色的下拉控件
       dw1.Items.Clear();
       for (int ii = 0; ii < count1; ii++)
         {
          dw1.Items.Add(fgcc[ii]);//给下拉控件增添项目
         }

       Label ls = (Label)row.FindControl("size1");
       string cs = ls.Text;            //取出数据表中的大小选项
       string[] fgcs = cs.Split(','); //以逗号为分隔符，将字符串分解为字符串数组
       int count2 = fgcs.GetLength(0);//确定数组的个数
       DropDownList dw2 = (DropDownList)row.FindControl("DropDownList2");
                  //取出选择的下拉控件
       dw2.Items.Clear();
       for (int ii = 0; ii < count2; ii++)
        {
          dw2.Items.Add(fgcs[ii]);  //给下拉控件增添项目
        }
       }
     }
```

29.2.5　生成购货车

在界面中选择参数后，单击【放入购货车】按钮，将生成购货车放入 Session ["ShoppingCart"]中，然后转到购货车网页。生成购货车的代码如下。

```
protected void Button1_Click(object sender, EventArgs e)
 {
  System.Data.DataTable Cart = new System.Data.DataTable();
  if (Session["ShoppingCart"] == null)
   {
     Cart.Columns.Add("产品编号", typeof(string));
     Cart.Columns.Add("购货量", typeof(int));
     Cart.Columns.Add("会员价", typeof(double));
     Cart.Columns.Add("颜色", typeof(string));
     Cart.Columns.Add("尺寸", typeof(string));
     Session["ShoppingCart"] = Cart;
   }
   Cart = (System.Data.DataTable)Session["ShoppingCart"];
   Label ProductIDControl = (Label)row.FindControl("Label2");
   string ProductIDText = ProductIDControl.Text;
   DropDownList QuantityControl =
(DropDownList)row.FindControl("DropDownList3");
   string QuantityText = QuantityControl.SelectedValue;
   int qq = int.Parse(QuantityText);
   Label UnitPriceControl = (Label)row.FindControl("UnitPriceLabel");
   string UnitPriceText = UnitPriceControl.Text;
```

```
    double pp = double.Parse(UnitPriceText);
    DropDownList ColorControl =
(DropDownList)row.FindControl("DropDownList1");
    string ColorText = ColorControl.SelectedValue;
    DropDownList SizeControl =
(DropDownList)row.FindControl("DropDownList2");
    string SizeText = SizeControl.SelectedValue;

    System.Data.DataRow rr = Cart.NewRow();
    rr["产品编号"] = ProductIDText;
    rr["购货量"] = qq;
    rr["会员价"] = pp;
    rr["颜色"] = ColorText;
    rr["尺寸"] = SizeText;

    Cart.Rows.Add(rr);
    Session["ShoppingCart"] = Cart;
    Response.Redirect("购货车.aspx");
   }
```

后面的汇总记账以及存放、查看订单等的方法与食品商店相同，这里不再重复。

29.3 账户管理

29.3.1 概述

账户是客户，但又不同于一般的客户，其不同点在于账户需要进行网上交易，因此应该拥有一定的权利(如允许访问一些特定的网页)，并提供更多的附加信息(如地址、联系人、联系电话、付费方式等)。如何给账户赋予这些权利并保存它们的附加信息，还要将这些附加信息与它们的基本信息紧密联系在一起呢？

有两种方式来保存账户的附加信息。一种是单独构建一套系统；另一种是利用系统已经提供的“基于角色的安全系统”或“个性化服务”并在这些基础上进行扩展。后一种方式优于前者，因为它能充分利用系统原有的安全体系，大大简化了设计过程。

下面将介绍后一种方式中的一种方案，这种方案要求建立两个模块(包含 4 张网页)和两个 Session 对象。

1. 账户注册模块

客户注册方法已经在 21.4.1 节与 21.4.2 节中讲述。现在要创建新账户，由于账户要求的信息更多，因此需要使用另一套 Login 和 CreateUserWizard 控件，并在该控件中增加新模板。

注意： 完成整个设置后，系统中可能会出现两套客户输入系统，其中一套用于输入一般客户和注册，另一套用于输入账户和注册。建议取不同的网页名字以避免混淆。

2. 账户验证模块

为了验证当前登录的客户是否为账户，需要增加两张网页(“账户判断.aspx”与“账户验证.aspx”网页)来进行判断。验证的依据主要看账户是否保存了“附加信息”。

3. Session 对象

将使用两个 Session 对象，其中 Session["zh_UserID "] 代表当前登录的客户名；Session["zh_dl"]代表是否为账户。

29.3.2　准备工作

1. 创建附加信息的数据表

账户的信息可以分为两部分：基本信息(如账户标记、密码等)，系统将会自动保存在自动产生的 ASPNETDB.MDB 数据表中；附加信息则根据需要另外设计一张表，例如表名为 CustomersTable。表结构如图 29.23 所示。

Bh	CustomerID	ContectName	Address	Phone	Premoney	ChangeDate
编号 (int)	账户标记 (nvarchar(30))	联系人 (nvarchar(20))	地址 (nvarchar(50))	电话 (nvarchar(30))	预付款 (money)	时间 (smalldatetime)

图 29.23　账户附加信息表

其中，Bh 为关键字，自动增加；CustomerID 是与基本信息相联系的字段；ChangeDate 为时间，将其默认值中设为 getdate()以便自动生成当前的时间。

2. 为账户准备角色

给网站分配角色的方法已经在第 21 章中讲述。其中也应该为账户确定一定的角色(例如 Customer)，并赋予一定的权限(如允许该角色访问结账、查看订单以及查看账户信息等)。如果在登录中使用了 LoginView 控件时，还应该给 Customer 角色分配相应的视图界面。

每个账户的角色是在创建时才赋予的(不需要预先赋予)。具体方法在后面讲述。

3. 给 Session 对象赋初值

在 Global.asax 文件的 Session_Start 事件中给 Session 对象赋初值。其代码如下。

```
void Session_Start(object sender, EventArgs e)
  {
    Session["zh_dl"] = null;
    Session["zh_UserID "] = null;
}
```

4. 编写存储过程

为了保存账户的附加信息，需要根据附加信息数据表的需要编写相应的存储过程。其代码如下。

```
Create PROCEDURE dbo.InsertCustomer
    (
    @CustomerID nvarchar(30),
    @ContectName nvarchar(20),
    @Address nvarchar(50),
    @Phone nvarchar(30)
    )
AS
```

```
    insert into [CustomersTable] (CustomerID,ContectName,Address,Phone)
Values (@CustomerID,@ContectName,@Address,@Phone)
RETURN
```

29.3.3 账户注册模块

在账户注册模块中要用到两张网页(“账户登录.aspx”与“创建新账户.aspx”)，网页中分别放入 Login 与 CreateUserWizard 控件。

1. 为 CreateUserWizard 控件增添新模板

账户注册时要用到 CreateUserWizard 控件。该控件由多个模板组成，默认情况下只包括“注册新账户”与“完成”两个模板，现在需要增添一个(或多个)新模板，以输入附加信息。

增添新模板的方法是，先将 CreateUserWizard 控件拖入网页，然后单击右方的【添加/移除 WizardSteps】项以添加新模板。然后在打开的【WizardStep 集合编辑器】对话框中，单击【添加】按钮，增加新模板并改变模板执行的顺序，将新增模板放在前两个模板之间，如图 29.24 所示。



图 29.24 WizardStep 集合编辑器

2. 给新模板进行布局

打开【添加账户信息】模板，利用表格进行布局。根据前面数据表的结构，放置 3 个 TextBox 控件以输入附加信息(还应该增添相应的校验控件，这里省略)。情况如图 29.25 所示。

3. 配置存储过程

配置数据源控件以满足调用存储过程的需要。方法是先将 SqlDataSource 数据源控件拖入模板，令其与 CustomersTable 表连接，配置调用存储过程(InsertCustomer)，参数则利用下列代码输入。

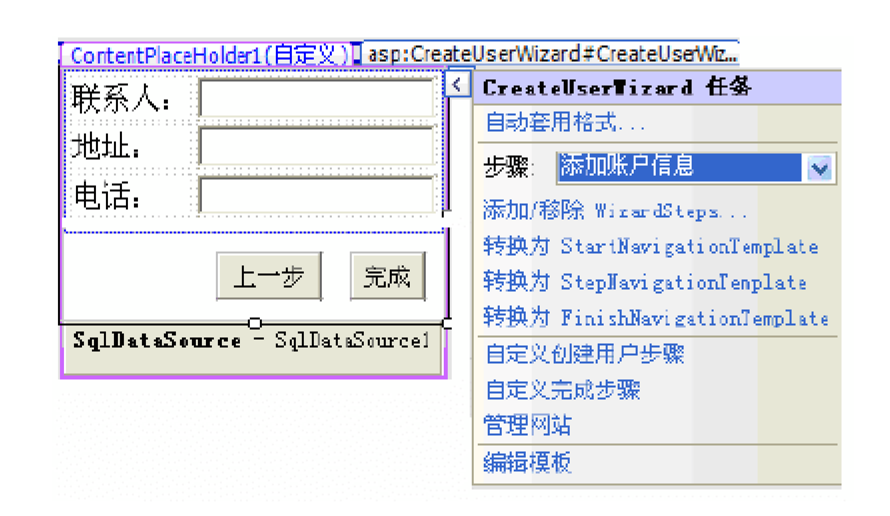


图 29.25　编辑账户附加信息模板

打开 CreateUserWizard 控件的属性对话框，双击 FinishButtonClick 事件，在事件中编写代码以输入存储过程需要的参数。

```
protected void CreateUserWizard1_FinishButtonClick(object sender,
WizardNavigationEventArgs e)
  {
   Session["zh_UserID "] = User.Identity.Name;
   SqlDataSource1.InsertParameters.Clear();
   SqlDataSource1.InsertParameters.Add("CustomerID",
Session["zh_UserID"].ToString());
   SqlDataSource1.InsertParameters.Add("ContectName", TextBox1.Text);
   SqlDataSource1.InsertParameters.Add("Address", TextBox2.Text);
   SqlDataSource1.InsertParameters.Add("Phone", TextBox3.Text);
   SqlDataSource1.Insert();
}
```

在上述代码中 User.Identity.Name 是 Page 类中一个重要的属性，该属性代表当前登录客户的名字(User Name)，利用它可以查询登录客户的各类信息。

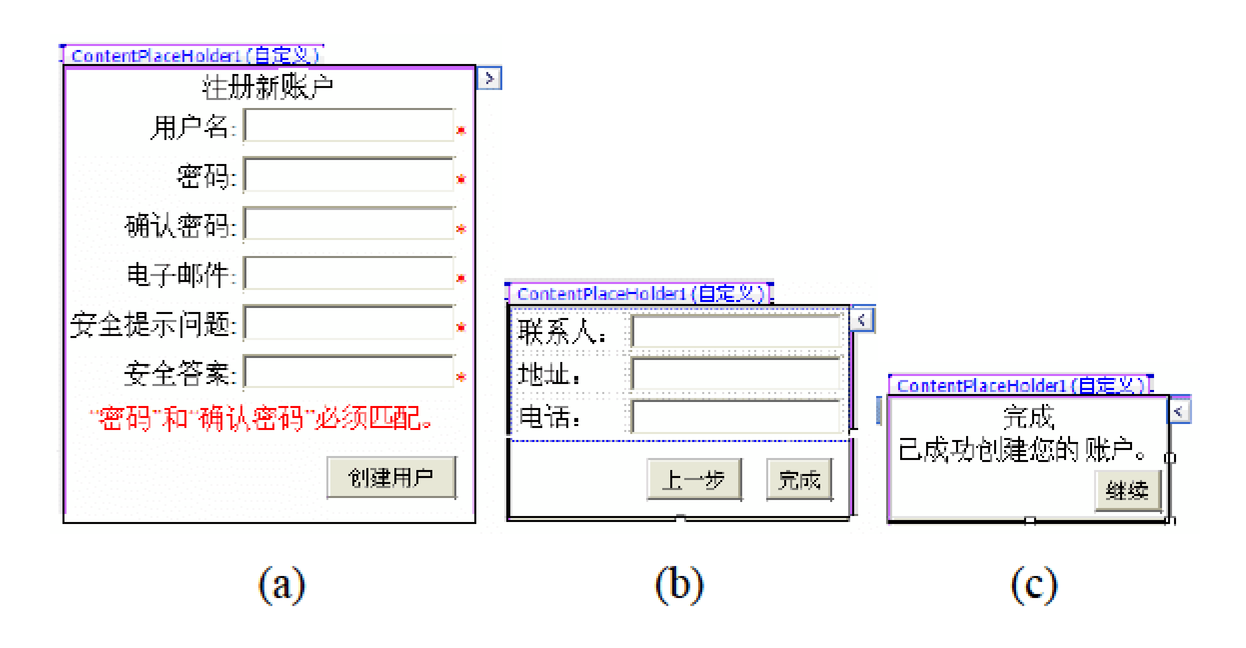
为了后面的需要，现在将其赋给 Session["zh_UserID "] 对象。即

```
Session["zh_UserID "] = User.Identity.Name;
```

从此 Session["zh_UserID "]也就代表当前登录的客户名了。

4. 三个模板的界面

最后三个模板的界面如图 29.26 所示。其中【注册新账户】模板就是输入客户基本信息的模板(见图 29.26(a))；【完成】模板中包括成功创建账户时的提示，以及【继续】执行时的按钮，如图 29.26(c)所示；【添加账户信息】模板如图 29.26(b)所示。



(a)　(b)　(c)

图 29.26　展开 CreateUserWizard 控件的多模板

5. 给新账户自动赋予角色

为了给新账户自动赋予角色，在 CreateUserWizard 控件的属性对话框中，双击 CreatedUser 事件，然后编写如下代码。

```
protected void CreateUserWizard1_CreatedUser(object sender, EventArgs e)
    {
     Roles.AddUserToRole((sender as CreateUserWizard).UserName, "Customer");
     }
```

注意：注意在网页代码的命名空间中要增加“using System.Web.Security;”的引用。

6. 转向“账户登录.aspx”网页重新进行登录

打开 CreateUserWizard 控件的属性对话框，双击 ContinueButtonClick 事件；然后编写以下代码，转向账户登录网页，以便重新登录账户。

```
protected void CreateUserWizard1_ContinueButtonClick(object sender,
EventArgs e)
 {
  Response.Redirect("账户登录.aspx");
 }
```

7. 返回“账户注册.aspx”网页

返回“账户注册.aspx”网页，在 CreateUserWizard 控件中，打开【注册新账户】模板，以便调用此网页时，控件从这个模板开始执行。这一点很重要，但也很容易被设计者忽略。

29.3.4 账户验证模块

为了判断客户是否注册，需增添两个网页：“账户验证.aspx”与“账户判断.aspx”。在账户验证.aspx 网页中放入一个 Gridview 控件通过数据源控件指向账户附加信息的数据表，设置数据源控件时，用 Session["zh_UserID"]对象作为查询参数。因为 Session["zh_UserID "]代表当前登录的客户名。而在注册表中，每位登录的客户都有一个唯一的名字，而且登录以后不允许修改。因此用这个名字查询不外乎两种结果。

- 没有找到相关记录(即查询结果为 0 行)，这说明登录的客户目前还没有注册附加信息，应转向“账户登录.aspx”网页，并提示客户注册。
- 找到了记录，这说明客户已经注册了附加信息，应将账户标记 Session["zh_dl"]设为 success (不再是 null)，并转向“账户判断.aspx”网页。

具体代码如下。

```
protected void Page_Load(object sender, EventArgs e)
    {
       Session["zh_UserID"] = User.Identity.Name;
       if (GridView2.Rows.Count == 0)  // 没有找到记录
       {
          Session["zh_dl"] = null;
          Response.Redirect("账户登录.aspx?action=您还没有进行账户注册。");
       }
```

```
        else
        {
            Session["zh_dl"] = "success";
            Response.Redirect("账户判断.aspx");  // 已经注册转向账户判断
        }
    }
```

29.3.5　账户管理的信息流程

为了让更多的客户浏览商品，在形成购货车之前不需要客户登录，直到打开购货车时才转入账户登录的信息流程。

图 29.27 是账户管理的信息流程示意图。其中各模块内部的连线比较容易理解，这里不作说明。下面重点讲述各模块之间的信息流通情况。

(1)　当客户选择完商品后，选择母版页中的菜单中的【购货车】项时，不直接打开购货车网页，而打开账户验证模块(见图中的①)。

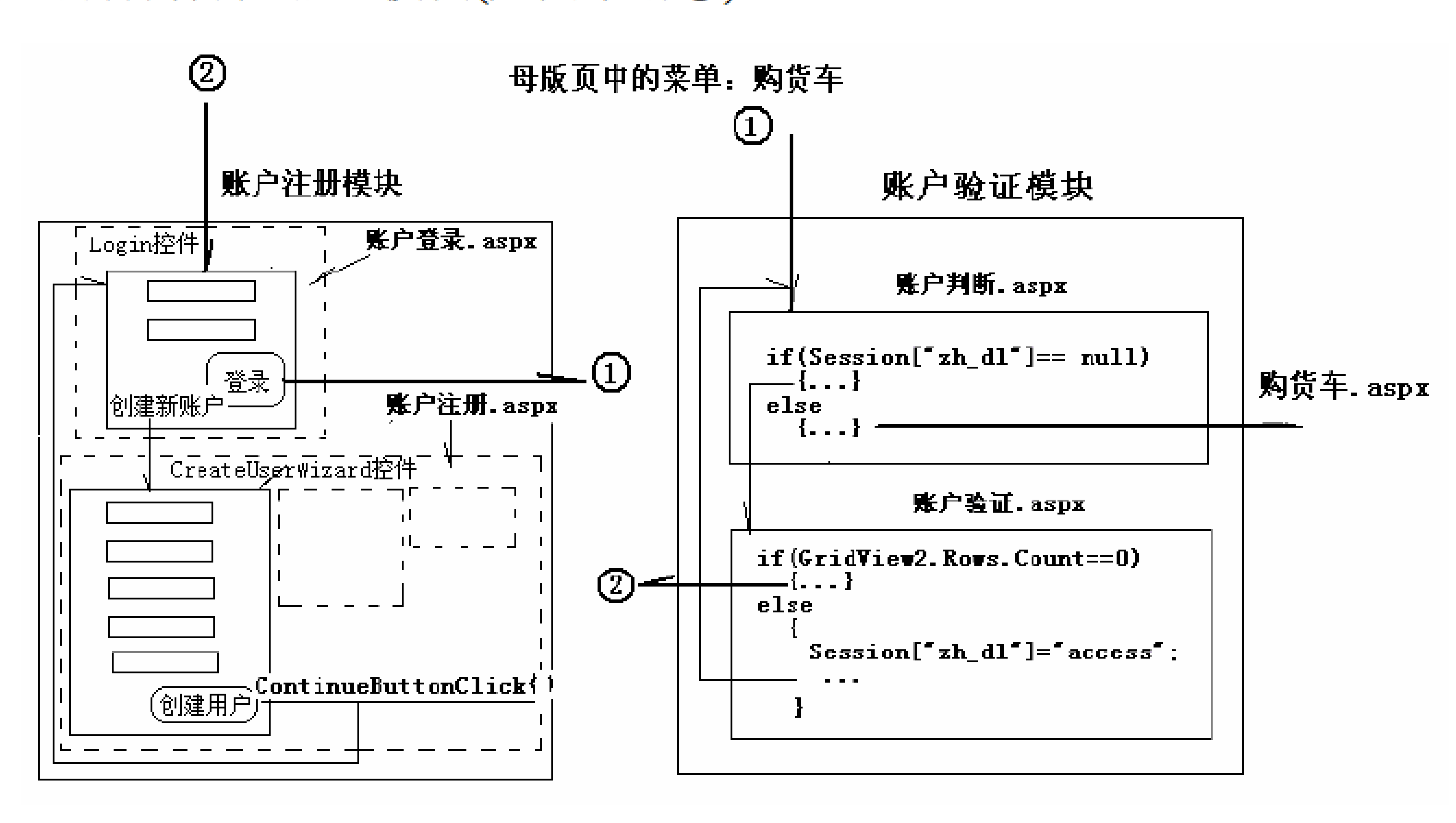


图 29.27　账户管理信息流程

(2)　如果该客户是账户时，直接转向“购货车.aspx”，否则转入“账户验证.aspx”。

(3)　在“账户验证.aspx”网页中，检查对账户附加信息表的查询结果，如果没有找到登录客户的信息时，转向账户注册模块，进行登录或注册(见图中的②)，否则将设置账户标记(Session[“zh_dl”])后，再转向“账户判断.aspx”网页重新进行判断。

(4)　在账户注册模块中完成注册并重新登录后，通过 Login 控件的【登录】按钮转向账户验证模块(见图中的①)。

(5)　在各个网页需要显示客户名字的地方，一律改用以下代码来实现。

```
TextBox1.Text = User.Identity.Name.ToString();
```

或者使用下列语句。

```
TextBox1.Text = (string)Session["zh_UserID "];
```

(6)　有一些网页不允许非账户访问，如“结账.aspx”、“订单.aspx”等。对于这些网页可以利用 Session 对象作进一步保护。代码如下。

```
void Page_Load(object sender, EventArgs e)
```

```
{
 if (Session["zh_dl"] == null)  //不是账户
   {
    Response.Redirect("账户判断.aspx?tt=您不能访问此网页.");
   }
   …
}
```

通过上述信息流程可以看出，账户第一次打开网站时，必须通过账户注册、账户登录才能进行网上交易。而以后再次打开网站时，只需在一般客户登录对话框或者账户登录对话框中输入账户名和密码即可进行网上交易。

29.4 小　结

电子商务是一种比较复杂的系统，各种电子商务有着不同的要求，涉及多种不同类型的网页，其中的变化也比较多。但是不论哪种电子商务，交易的过程都要包括选购、结账、存储订单、查看订单等几个基本阶段。本章利用 SQL Server 提供的 Northwind 样板库中的数据，讲解了这几个基本阶段的设计工作。理解本章的关键在于搞清 GridView、内存数据表和 Session 对象三者之间的关系。GridView 提供了数据库中的原始数据；内存数据表是动态生成的，存储了客户选择的数据，它是数据库部分数据的副本(另外还增添了一些临时数据)；Session 对象代表购货车，用来将数据进行保存，并在不同的网页中共享。利用内存数据表存储数据，利用 Session 传递数据，这种处理方式减少了访问数据库的次数，提高了程序运行的效率。

示例中多次用到 IsPostBack 这个属性，这个属性用来区分网页是第一次打开还是事件处理完毕后再次被打开这两种情况。当网页第一次被打开时该属性为 false；如果是在事件处理完返回本页时其值为 true；在一定条件下该属性发挥了重要的作用。例如语句

```
if (!IsPostBack)
{
//以 Session 作为数据源
}
```

代表只有第一次打开网页(不是后续打开)时，才执行“以 Session 作为数据源”的操作，至于后续打开网页时的数据，则是靠视图状态保持下来的。详细情况参考第 9 章。

为了使得商品的显示界面更加生动，后面介绍了将表格与图像相结合的设计方法，关键是使表格网页与图像网页进行同步，并在图像网页中，利用 GridView 控件的模板进行布局并进行数据绑定。

除此而外，在本章后面还对服装的电子商务设计进行了讨论。与食品(还有书籍等)商店不同，这些网站的外观选择更为重要，还可能包括更多的选择参数，需要动态生成一些控件。利用 ListView 控件的平铺布局设计主界面，能够更好地突出外观图像；利用自动生成的下拉控件可以根据数据表中的数据提供更多的可选参数。

对账户的管理也是电子商务中不可缺少的部分，为了获得账户的附加信息，可以在 CreateUserWizard 控件中增添模板，这样做的好处是，可以将这些附加信息与基本信息联系起来，而且还能借用系统的角色保护功能。

User.Identity.Name 是 Page 类中一个重要属性。它代表当前登录客户的账号(User Name)，利用它可以查询当前登录客户的信息，并且判断该客户是否已经进行了注册。

29.5 习 题

1. 填空题

(1) 下面是本章应用程序中生成购货车的部分代码。填写带字母标号语句的含义。

```
void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
 {
  System.Data.DataTable Cart = new System.Data.DataTable();    //A
  if (e.CommandName == "buy")
   {
    if (Session["ShoppingCart"] == null)                       //B
     {
      Cart.Columns.Add("商品编号", typeof(int));
      Cart.Columns.Add("商品名称", typeof(string));
      …
      Session["ShoppingCart"] = Cart;
     }
    Cart = (System.Data.DataTable)Session["ShoppingCart"];     //C
   if (TextBox2.Text == "")
    {
    Validate();                                                //D
    }
   else
   {
    int index = Convert.ToInt32(e.CommandArgument);
    GridViewRow row = GridView1.Rows[index];
    TextBox tt =
(System.Web.UI.WebControls.TextBox)row.Cells[1].FindControl("TextBox
    1");                                                       //E
string dgl = tt.Text;
    int dg = int.Parse(dgl);
    if (dg <= 1) dg = 1;
    string djText = row.Cells[5].Text;
double dj = double.Parse(djText);                              //F
    System.Data.DataRow rr = Cart.NewRow();                    //G
    rr["商品编号"]=bh;
   …
Cart.Rows.Add(rr);
    Session["ShoppingCart"] = Cart;                            //H
   }
  }
 }
```

A ____

B ____

C ______________________________

D ______________________________

E ______________________________

F ______________________________

G ______________________________

H ______________________________

(2) 下面是本章应用程序中利用 GridView 控件进行汇总计账的代码。填写带字母标号语句的含义。

```
void Button2_Click(object sender, EventArgs e)
 {
   double sum=0.0;
   for(int ii=0; ii < GridView1.Rows.Count;ii++)                //A
{
    CheckBox cc =
(CheckBox)GridView1.Rows[ii].Cells[0].FindControl("CheckBox1");//B
     if(cc.Checked)                                              //C
      {
      sum = sum + (double.Parse(GridView1.Rows[ii].Cells[7].Text));
                                                                 //D
      }
    }
   TextBox2.Text=sum.ToString();                                 //E
}
```

A ______________________________

B ______________________________

C ______________________________

D ______________________________

E ______________________________

(3) 为了利用 productID 进行跨页同步，代码是:

```
<a href = "另一网页.aspx?id=______________________________"> 同步 </a>
```

(4) 为通过 Label 控件取出数据表中 color 字段的数据，但不显示 Label 控件时的代码是

```
<asp Label ID = "…" runat = "server" Text='_______________'
       Visible="______" />
```

(5) 设 ss 是一个字符串(中间包括若干句号“.”)，现在要将其以句号为界分解成字符串数组，其代码是

```
string[] sz = _________________;
```

2. 选择题

(1) 内存数据表在本应用系统中的作用是_______________。

A. 网页间共享数据　　　　B. 汇集并临时存存记录

C. 临时保存数据　　D. 保存和传送数据

(2) Session 对象在创建虚拟购货车中的作用是______________。

A. 保存数据　　B. 汇集记录

C. 临时保存数据　　D. 网页间保存和传送数据

(3) Page.IsPostBack 属性在______________情况下为 true。

A. 新打开网页时　　B. 返回本窗体时

C. 关闭网页时　　D. 退出网页时

(4) User.Identity.Name 属性代表______________。

A. 新增添的客户名　　B. 管理员客户名

C. 当前登录的客户名　　D. 具有特权的客户名

3. 简答题

(1) 在购货车的创建中 Session 对象起什么作用?

本章的应用程序中下列代码起什么作用?

```
if(!IsPostBack)
{
  GridView1.DataSource= Session["ShoppingCart"];
  DataBind();
}
```

(2) 在执行了上述代码的情况下，后续网页中 GridView 控件中的数据是怎么得来的?

(3) 账户管理中如何获得附加信息并与客户登录的基本信息联系起来?

(4) 如何判断目前登录的客户是否已经注册?

4. 操作题

(1) 按照本章所讲的步骤完成选购、计账、下订单等项工作。

(2) 在完成上述工作的基础上增加下述附加功能:

- 允许客户随时查看产品的出产单位。
- 允许特权客户修改产品的单价。
- 允许特权客户查看和修改雇员的情况。
- 实现从客户的预付款中扣除货款的功能。
- 设计一个表格与图像相结合的商品介绍界面。
- 动态生成控件以提供更多的选项。

(3) 设计一个客户注册系统，系统中的某些网页只允许已经注册的客户访问。若客户没有注册就想打开时，将给予适当的提示并返回到注册表输入网页。

(4) 设计一个服装(或首饰、花卉)网站。

附录 A　C#.NET 基础语法参考

C#.NET 语言由微软公司开发(并得到 ECMA 与 ISO/IEC 标准的承认)，是一种现代的、简单的、面向对象的、通用的程序设计语言，从 2002 年到 2010 年已经先后开发了 5 个版本(1.0、1.2、2.0、3.0、4.0)，本附录作为 C#.NET 的基础语法参考。

A.1　数 据 类 型

A.1.1　常量与变量

常量即在程序的运行过程中其值不会发生改变的量。常量声明的格式为：

```
常量修饰符　const　类型标识符　常量名＝常量表达式;
```

其中常量修饰符可以是 new、private、protected、internal 和 public。

常量定义举例如下。

```
public const int x=1,y=3;
```

变量是在程序的运行过程中其值可以发生改变的量。从用户的角度来看，变量就是存储信息的基本单元；从系统的角度来看，变量就是计算机内存中的一个存储空间。变量必须先定义后使用。

变量定义的一般形式为：

```
[变量修饰符] 类型标识符　变量标识符[=常量表达式];
```

变量修饰符有 new、private、protected、internal、public、static 和 readonly。

变量定义举例如下。

```
static public int x=1;
```

A.1.2　值类型

1. 简单类型：整型、实型、字符型和布尔型

1)　整型

整型变量的值为整数。C#中有 8 种整数类型：短字节类型 sbyte、字节型 byte、短整型 short、无符号短整型 ushort、整型 int、无符号整型 uint、长整型 long、无符号长整型 ulong。各类型的说明及取值范围如表 A.1 所示。

表 A.1　整型变量

数据类型	说　明	取值范围
sbyte	有符号 8 位整数	−128～127
byte	无符号 8 位整数	0～255

续表

数据类型	说　明	取值范围
short	有符号 16 位整数	−32 768～32 767
ushort	无符号 16 位整数	0～65 535
int	有符号 32 位整数	-2^{31}～$2^{31}-1$
uint	无符号 32 位整数	0～2^{32}
long	有符号 64 位整数	-2^{63}～$2^{63}-1$
ulong	无符号 64 位整数	0～$2^{64}-1$

2)　实型

实型数据分为浮点型和十进制类型。

数学中的小数在 C#中采用两种数据类型来表示：单精度(float)和双精度(double)。它们的取值范围和精度如下。

- 单精度：取值范围为$\pm1.5\times10^{-45}$～$\pm3.4\times10^{38}$，精度为 7 位数。
- 双精度：取值范围为$\pm5.0\times10^{-324}$～$\pm1.7\times10^{308}$，精度为 15～16 位数。

十进制类型(decimal)是 C#专门提供的一种类型，主要用于金融和货币方面的计算。该类型是一种高精度、128 位数据类型，运算结果准确到小数点后 28 位。当定义一个 decimal 变量并赋值给它时，使用 m 下标以表明它是一个十进制类型，例如，decimal dx=1.0m;。

3)　字符型

字符包括数字字符、英文字母、表达符号等，C#提供的字符按照国际公认的标准，采用 Unicode 字符集。一个 Unicode 的标准字符长度为 16 位。C#中也存在转义字符，用来在程序中指代特殊的控制字符。C#中使用的转义字符如表 A.2 所示。

表 A.2　转义字符

转义字符	字符名称
\’	单引号
\”	双引号
\\	反斜杠
\0	空字符(Null)
\a	响铃
\b	退格
\f	进纸
\n	换行
\r	回车
\t	水平制表
\v	垂直制表

除了简单的转义字符外，可以通过 C#的十六进制转义序列来表示任何一个 Unicode

字符常数。例如，'\x101'等价于字符'A'。

2. 结构类型

结构类型是指把各种不同类型数据信息组合在一起形成的类型。结构类型的变量采用 struct 来声明。例如可以定义学生成绩表记录结构如下。

```
struct Student
{
public string number;
public string name;
public int score;
}
Student stu1;
```

stu1 就是一个 Student 结构类型的变量。对结构成员的访问通过结构变量名加上访问符“.”号，再跟成员名，如 stu1.name="Jacky";。

结构类型包含的成员类型没有限制，结构类型的成员还可以是结构类型。

3. 枚举类型

枚举实际上是为一组在逻辑上密不可分的整数值提供便于记忆的符号。

例如，声明一个代表颜色的枚举类型的变量。

```
enum Color {red,yellow,blue,green,black,white};
Color col;
```

枚举类型的变量在某一时刻只能取枚举中某一个元素的值。例如 col 这个表示颜色的枚举类型变量，在某一时刻只能为枚举中的一种颜色。

枚举中的每一个元素类型都是 int 型，第一个元素的值为 0，其后每一个连续的元素依次加 1 递增。也可以给元素直接赋值，如把 red 的值设为 1，其后元素的值分别为 2、3、…

```
enum Color {red=1,yellow,blue,green,black,white};
```

为枚举元素所赋的值的类型限于 long、int、short 和 byte 等整数类型。

A.1.3　引用类型

C#中的另一大数据类型为引用类型，引用类型与值类型的区别在于：引用类型变量不直接存储所包含的值，而是实际数据的地址。C#中的引用类型有 4 种：类、数组、代理和接口。

1. 类

类是面向对象编程的基本单位，其中包含数据成员、函数成员和嵌套类型的数据结构。类的数据成员有常量、字段和事件。函数成员包括方法、属性、索引指示器、运算符、构造函数和析构函数。类支持继承机制，派生类可以扩展基类的数据成员和函数成员。

属性是 C#.NET 语言中提出的一种新的成员。为了保护数据，通常将数据成员定义成

私有(private)保护等级，将其封装在类中；同时，为了方便外界的访问，又必须提供数据对外访问的接口。属性的使用简化了私有数据对外访问接口的设置过程。通常属性中包括两个专用的方法，一个用于读取私有数据；另一个用于写入或修改私有数据。

属性的定义格式如下。

```
[修饰符] 返回类型 属性名
{
  get      // 用于读取私有数据
  { …}
  set      // 用于写入私有数据
  { …}
}
```

其中修饰符可以使用 public、protected、private、internal 等，通常使用 public。

当与私有变量结合使用时，属性定义的代码如下。

```
private 返回类型  私有变量名;              //定义一个私有变量
public 返回类型  属性名                    //属性定义开始
  {
   get                                     //读取私有变量
    { return  私有变量名 ; }
  set                                      //写入私有变量
    { 私有变量名  =  value ; }
}                                          //属性定义结束
```

其中 get 方法用于读取成员变量，set 方法用于写入或改变成员变量。当从属性读出数据时，系统将自动调用 get 方法；当给属性赋值时系统将自动调用 set 方法。如果属性中只包括 set 方法时，此属性只能写入；只包括 get 方法时，此属性为只读属性。为了保护数据，可以在读取或设置成员变量的语句前面加上条件判断，当条件满足时才能对变量进行读或写操作。

属性的定义及调用举例如下。

(1) 定义属性：

```
class Employee
{
  private string name;       //定义 1 个私有变量
  public string Name
 {                           //定义 Name 属性
  get
   { return name; }
  set
   { name = value; }
 }
}
```

(2) 调用属性：

```
Employee e1 = new Employee();
e1.Name = "Cheng";              //自动调用属性 Name 中的 set 方法
Console.WriteLine (e1.Name);    //自动调用属性 Name 中的 get 方法
```

从上例可以看出，调用属性的格式与调用变量的格式相同，只不过用属性名代替了变

量名。

2. 数组

数组是同一类型数据的有序集合。数组的声明格式如下。

```
数组类型[]  数组名;
```

声明一个整数数组：

```
arr: int[]  arr;
```

使用数组元素时，可以用数组名[下标]来获取对应的数组元素。C#中的数组元素的下标从 0 开始，以后逐个加 1。C#中的数组可以是一维的，也可以是多维的。在数组声明的时候可以对数组元素进行初始化。一维及多维数组的定义和初始化举例如下。

```
class Class1
{
static void Main()
{   string[] a1=new int[] {1,2,3};          //一维数组
string[,] a2=new int[,] {{1,2,3},{4,5,6}}; //二维数组
string[,,] a3=new int[10,20,30];           //三维数组
}
}
```

3. 代理

面向对象应用程序中，对象与对象之间通过消息和事件建立联系。当一个对象发生了某种事件时，会发出一定的消息去引发其他对象的方法来响应该事件。如何在事件源与接收方之间建立联系呢？.NET 框架提供了一种特殊的类型“代理(Delegate)”，在事件源与事件接收者之间架起了一座桥梁。

代理的作用与结构化程序中函数指针的作用相似，但具有面向对象、类型安全等新的特征。代理是一个派生于 System.delegate 的类，通过“方法签名”来识别它所引用的方法。一个方法的参数类型(包括它们的顺序)与返回类型的组合就称为该方法的签名。代理只能对与代理定义中的签名相匹配的方法进行引用。

定义代理的格式如下。

```
[ accessModifier ]delegate delegateName returnType ( [parameterType
parameterName [,…]] )
```

其中 delegate 是关键字。其他部分的含义如下。

- accessModifier：对代理的存取权限，可以定义成 public、protected 等。
- delegateName：代理类的名字。
- returnType：代理类返回的类型。
- parameterType：传递给代理的参数类型。
- parameterName：传递给代理的参数名。

例如，定义一个代理 Caculation。

```
public delegate double Caculation(double x, double y);
```

这个代理有两个 double 类型的参数，返回一个 double 类型的值。这个代理只能将信息传递给有两个 double 参数，并且返回 double 类型的方法。

将代理与实例方法连接的程序举例如下。

```
using System;
namespace Delegate2
{
    public delegate string DelegateString();
    public class Student
    {
     private string name;
     private int age;
     public Student(string name, int age)
{
         this.name = name;
         this.age = age;
      }
     public string NameAndAge()
{
       return(name + " is " + age + " years old");
      }
    }
    public class Speciality
    {
      private string department;
      private string special;
      public Speciality(string dep,string spe)
      {
        this.department = dep;
        this.special = spe;
      }
     public string show()
      {
         return("The department is "+department +" special is "+special);
      }
     }
    class Test
    {
     static void Main(string[] args)
     {
      Student stu1 = new Student("Cheng",19);
      DelegateString myDelegate1 = new DelegateString(stu1.NameAndAge);
      string st = myDelegate1();
      Console.WriteLine(st);
      Speciality sp = new Speciality("Computer","Application");
      DelegateString myDelegate2 = new DelegateString(sp.show);
      string spec = myDelegate2();
      Console.WriteLine(spec);
    }
  }
}
```

该程序中将代理对象与方法连接的代码有。

```
Student stu1 = new Student("Cheng",19); //生成 Student 类的对象 stu1
DelegateString myDelegate1 = new DelegateString(stu1.NameAndAge);
    //对象 myDelegate1 与方法 NameAndAge()连接
Speciality sp = new Speciality("Computer","Aplication");
    //生成 Speciality 类的对象 sp
DelegateString myDelegate2 = new DelegateString(sp.show);
    //对象 myDelegate2 与方法 show()连接
```

通过代理引用方法的语句有

```
string st = myDelegate1();// 引用方法 NameAndAge()
Console.WriteLine(st);
string spec = myDelegate2();// 引用方法 show()
Console.WriteLine(spec);
```

因此该程序运行结果为

```
Cheng is 19 years old
The department is Computer special is Application
```

4. 接口

接口中不能定义数据，只能定义方法(包括方法、属性、索引指示器等)，而且只能定义方法的首部，不包括这些方法的实现。在接口中所定义的实际上是一组为所有继承者必须遵守的契约。对于使用者来说，只要知道某个类是从哪个接口继承的，就知道这个类能提供什么样的服务，以及如何调用这些提供服务的方法。

接口与接口之间可以继承，类(或结构)也可以继承接口。类与类之间只能单端继承，但类可以在继承一个基类的同时继承一个或多个接口。在继承接口的类中必须实现全部在接口中定义的方法。

接口的声明要用到 interface 关键字。声明的基本格式如下。

```
[修饰符] interface 接口名称 [: 基接口表 ]
{
  // 声明接口成员
}
```

例如：

```
    using System;
    interface IPoint
{
void Print(Object o);
int X {get; set;}
int Y { get; set;}
     }
```

下面通过一个简单的例子来说明接口的使用。假设接口 IPoint 的声明同上例，首先定义一个继承自该接口的类 myPoint 如下。

```
public class myPoint : IPoint  // 类 myPoint 继承自接口 IPoint
{
    private int x;
    private int y;
```

```
        public myPoint(int x, int y)
        {
            this.x = x;
            this.y = y;
        }
        public void Print(Object o)         //实现接口中定义的方法
        {
            IPoint tp = (IPoint)o;
            Console.WriteLine("点坐标X:{0},Y:{1}",tp.X,tp.Y);
        }
        public int X                        //实现接口中定义的属性
        {
            get { return x; }
            set { x=value; }
        }
        public int Y                        //实现接口中定义的属性
        {
            get { return y; }
            set { y=value; }
        }
        public static void Main()
        {
            myPoint p = new myPoint(10,20);
            p.Print(p);
        }
    }
```

运行结果为：

```
点坐标 X:10, Y:20
```

A.1.4 装箱与拆箱

装箱和拆箱机制使得在 C#类型系统中，数值类型的变量与引用型变量之间可以互相转换。

装箱(Boxing)是指把数值类型的变量转换成引用类型的操作。把一个值类型装箱，就是创建一个 object 类型的实例并把该值类型数据复制给这个 object 实例。

装箱操作举例如下。

```
int i=2000;
object  obj=i;
```

装箱操作转换过程如图 A.1 所示。

将引用类型转换为数值类型的操作称为拆箱(Unboxing)，该操作必须显式地完成。拆箱操作举例如下。

```
object obj1;
int j = (int) obj1;
```

拆箱操作转换过程如图 A.2 所示。

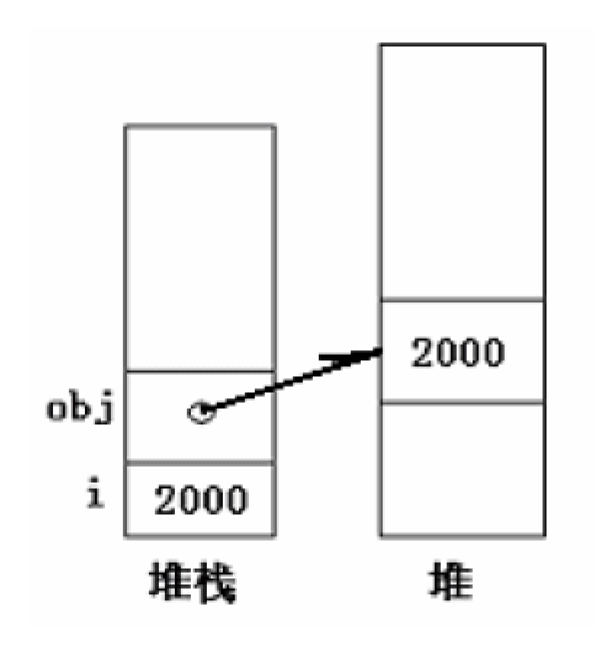



图 A.1　装箱操作转换过程

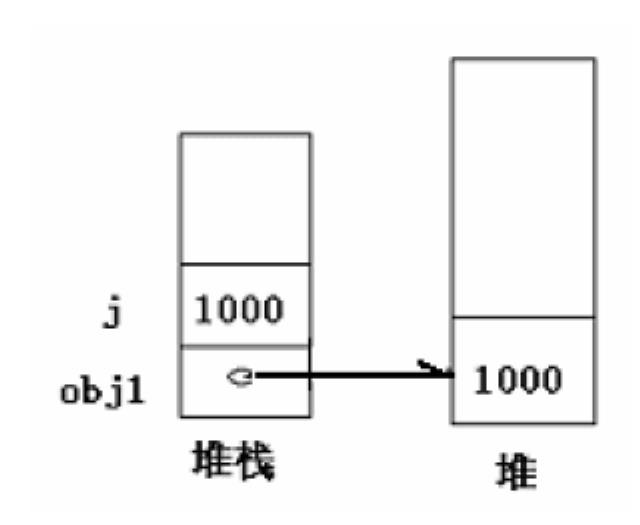


图 A.2　拆箱操作转换过程

A.2　运算符与表达式

表达式由运算数和运算符组成，表达式中的运算符指出了对操作数的操作。

A.2.1　算术运算符与算术表达式

算术运算符用来执行常规的算术运算，包括+(加)、−(减)、*(乘)、/(除)和%(模，求余数)。

A.2.2　赋值运算符与赋值表达式

赋值运算符主要用来进行赋值操作，包括=、+=、−=、*=、/=、%=、=、|=、^=、>>=、 <<=。复合赋值运算符将左操作数与右边表达式进行相应的运算，然后把结果赋给左操作数。例如，x+=6 等价于 x=x+6。

A.2.3　关系运算符与关系表达式

关系运算符主要进行表达式的比较操作，包括=(等于)、！ =(不等于)、>(大于)、<(小于)、>=(大于等于)和<=(小于等于)。这些运算符都是二元运算符，运算结果为布尔值 true 或 false。

A.2.4　逻辑运算符与逻辑表达式

逻辑运算符用来执行逻辑运算，包括&&、||和!，其中&&、||都是二元运算符。各运算符的运算规律如表 A.3 所示。

表 A.3　逻辑运算符的真值表

x	y	!x	x&&y	x||y
true	true	false	true	true

续表

x	y	!x	x&&y	x‖y
true	false	false	false	true
false	true	true	false	true
false	false	true	false	false

如果表达式中同时存在着多个逻辑运算符，逻辑非的优先级最高，逻辑与的优先级高于逻辑或。

用逻辑运算符将关系表达式或布尔值连接起来就是逻辑表达式。逻辑表达式的值仍然是一个布尔值。例如，给出一个年份，要判断它是不是闰年，设年份为 year，是否闰年可以用以下逻辑表达式来表示。

```
(year%400==0)|| (year%4)==0&&(year%100!=0)
```

A.2.5 位运算符

位运算符用来对数据按二进制位进行运算，包括&(与)、|(或)、^(异或)、~(取反)、<<(左移)和>>(右移)。其中取补是一元运算符，其他操作符都是二元运算符。位运算的操作数为整型数或者是可以转换为整型的任何其他类型。

A.2.6 对象创建运算符

对象创建运算符即 new 运算符，用来创建对象并调用对象的构造函数进行初始化。例如，

```
Class1 obj=new Class1( );
```

该语句用于创建类 Class1 的对象 obj，并调用 Class1 的构造函数 Class1()对该对象进行初始化。

A.2.7 其他运算符

下面介绍其他运算符。

- ++递增运算符：对操作数进行加法运算。例如，语句 i++；等价于语句 i=i+1。
- --递减运算符：对操作数进行减法运算。例如，语句 i--；等价于语句 i=i−1。
- 条件运算符：“?:”，是一个三元运算符。使用的一般形式为(表达式 1)? (表达式 2):(表达式 3)，先求解表达式 1 的值，如果表达式 1 的值为 true，将表达式 2 的值作为整个表达式的值，否则将表达式 3 的值作为整个表达式的值。例如，

  ```
  int max=(x>y)?x: y;
  ```

如果 x>y 为 true，则把 x 赋给 max，否则，把 y 赋给 max。

A.3　流程控制语句

C#提供了以下流程控制语句。

- 分支控制语句：if 语句和 switch 语句。
- 循环控制语句：while 语句、do-while 语句、for 语句和 foreach 语句。
- 跳转语句：break 语句和 continue 语句。
- 异常处理语句：try 语句、catch 语句和 finally 语句。

A.3.1　分支控制语句

当程序中需要进行两个或两个以上的选择时，可以根据条件判断来选择将要执行的一组语句。C#提供的选择语句有 if 语句和 switch 语句。

1. if 语句

if 语句是最常用的选择语句，用来判断是否满足给定的条件，根据判定的结果决定执行给出的两种操作之一。

if 语句有两种基本形式。

1)　if （布尔表达式）语句

当布尔表达式的值为真，则执行 if 后面的内嵌语句，为假，则继续执行 if 语句的后继语句。

例如，对一个浮点数 x 求绝对值，结果保存在 x 中。

```
if  (x<0) x= -x;
```

2)　if (布尔表达式) 语句 1 else 语句 2

当布尔表达式的值为真，则执行内嵌语句 1，否则执行内嵌语句 2。

例如，将两个整型数 x、y 中较大的赋给整数 max。

```
if  (x>y) max=x; else max=y;
```

如果程序的逻辑判断关系比较复杂，通常还可以采用条件判断嵌套语句。具体形式如下。

```
if  (布尔表达式)
{if (布尔表达式) 语句1  else 语句2}
else
{if (布尔表达式) 语句3  else 语句4}
```

此时每一条 else 与离它最近且没有其他 else 与之对应的 if 相搭配。

例如，一数学函数的表达式为

$$y=\begin{cases}-1 & x<0\\ 0 & x=0\\ 1 & x>0\end{cases}$$

使用 if 语句书写代码如下。

```
if (x>0) {y=1;}
else {if (x==0) y=0;else y=-1;}
```

2. switch 语句

如果要实现多分支语句，可以采用 switch 语句。switch 语句根据一个控制表达式的值选择一个内嵌语句的分支来执行。它的一般格式为

```
switch(控制表达式)
{case 常量表达式 1: 语句 1 break;
case 常量表达式 2: 语句 2 break;
            ⋮
case 常量表达式 n: 语句 n break;
default: 语句 n+1
}
```

switch 语句的语义为：当控制表达式的值与某一个 case 后面的常量表达式的值相等时，就执行此 case 后面的语句，若与所有的 case 中的常量表达式的值都不能匹配时，就执行 default 后面的语句。C#语言规定：switch 语句中每一个 case 标签后面必须使用 break 语句或跳转语句 goto，不允许从一个 case 标签贯穿到另一个 case 标签。

switch 语句使用示例要求按照考试成绩的等级(Grade)打印出百分制分数段。

```
switch(grade)
{case 'A': printf("85～100\n"); break;
 case 'B': printf("75～84\n"); break;
 case 'C': printf("60～74\n"); break;
 case 'D': printf("<60\n"); break;
 default: printf("error\n");
}
```

A.3.2 循环控制语句

循环语句用于实现一个程序模块的重复执行。C#提供了 4 种循环控制语句：while 语句、do-while 语句、for 语句、foreach 语句。

1. while 语句

while 语句用来实现“当型”循环结构。其一般形式如下。

```
while (布尔表达式) 语句
```

它的执行顺序为：

(1) 计算布尔表达式的值。

(2) 当布尔表达式为真时，执行 while 语句中的内嵌语句，然后程序转至第(1)步。

(3) 当布尔表达式为假时，结束循环。

例如用 while 语句求 1+2+3+⋯+100，代码如下。

```
i=1;
sum=0;
while(i<=100)
```

```
{sum+=i;
i++;}
printf("1+2+3+…+100=%d",sum);
```

2. do-while 语句

do-while 语句的特点是先执行循环体，然后判断循环条件是否成立。其一般形式为：

```
do
循环体语句
while(布尔表达式)
```

该语句执行顺序如下。

(1) 执行循环体语句一次。

(2) 计算布尔表达式的值，为真则回到第一步，为假则终止 do 循环。

例如用 do-while 语句求 1+2+3+…+100，代码如下。

```
i=1;
sum=0;
do
{sum+=i;
i++;}
while(i<=100);
printf("1+2+3+…+100=%d",sum);
```

3. for 语句

在事先知道循环次数的情况下，使用 for 语句比较方便。for 语句的格式为：

```
for(表达式 1；表达式 2；表达式 3) 语句
```

其中表达式 1 为循环控制变量初始化，循环控制变量可以有一个或多个，若有多个则用逗号隔开；表达式 2 为循环控制条件；表达式 3 按规律改变循环控制变量的值。3 个表达式都是可选的，缺省某个表达式时，其后的分号“；”不能省，如三个表达式都省略的情况如下。

```
for(； ； ){语句}
```

for 语句的执行顺序如下。

(1) 按书写顺序将表达式 1 执行一遍，为循环控制变量赋初值。

(2) 测试表达式 2 是否为真。

(3) 若没有表达式 2 或表达式 2 为真，则执行内嵌语句一遍，按表达式 3 的规律改变循环控制变量的值，回到第(2)步执行。

(4) 若表达式 2 不满足，则 for 循环终止。

例如用 for 语句计算 1+2+3+…+100，代码如下。

```
for(i=1;i<=100;i++) sum+=i;
```

4. foreach 语句

foreach 语句是 C#中新引入的，它表示收集一个集合中的各元素，并针对各元素执行内嵌语句。foreach 语句的一般形式为：

```
foreach(类型标识符  标识符 in 表达式) 语句
```

其中类型标识符和标识符用来声明变量，表达式对应集合。每执行一次内嵌语句，循环变量就依次取集合中的一个元素代入其中。

例如用 foreach 语句输出整型数组 arr 中所有元素，代码如下。

```
int[] arr=new int[] {1,2,3,4,5};
foreach(int a in arr)
Console.WriteLine(a);
```

A.3.3 异常处理语句

C#中提供的异常处理语句有 try、catch、finally。

try 语句提供了一种机制来捕捉块执行过程中发生的异常。它的一般形式为：

```
try
{被保护的语句块}
catch(异常声明 1)
{语句 1}
catch(异常声明 2)
{语句 2}
    ⋮
catch
{语句 n}
```

该语句的执行过程为：首先执行 try 语句中被保护的语句块，如果发生异常，则根据异常类型与其后 catch 块中的异常声明匹配，如果与某一个 catch 块中的异常声明匹配，则执行其内嵌语句，如果与任何一个 catch 块中的异常声明都不匹配，则执行不带异常声明的 catch 块中的语句。

finally 语句块通常用来完成一些善后工作，即不论是否发生异常都必须执行的语句块放在 finally 语句块中。

try-catch-finally 语句块的使用举例如下：从键盘输入一整型数作为除数，当除数为 0 时将抛出并处理异常。

```
class Class1
    {
        private int x=100;
        public int div(int ii)
        {
         int y;
         try
            {
             y = x / ii;
            }
         catch (DivideByZeroException) // 除数为 0 时的处理
         {Console.WriteLine("除数为零，请重新输入！\n"); y=0;}
        finally
        {Console.WriteLine("执行 finally 语句块\n");}
         return y;
```

```
        }
        static void Main(string[] args)
         {
            int s,s1;
            Class1 dd=new Class1();
            s = Console.Read() - 48;  // 0 的 Unicode 代码为 48
            s1=dd.div(s);
          Console.WriteLine("结果为：{0}",s1);
         }
    }
```

执行该程序时，若输入整数 0，将引发 Divide By Zero Exception 异常，所以将执行 catch 块中的语句，输出结果为：

```
除数为零，请重新输入！
执行 finally 语句块
结果为：0
```

附录 B　部分习题参考答案

第 1 章

1. 填空题

(1) CGI 阶段，脚本语言阶段，组件技术

(2) 程序设计语言，应用程序平台，ADO.NET 及类库，公共语言运行库，Visual Studio.NET 开发环境

(3) 命名空间

2. 选择题

(1) C　(2) D　(3) C　(4) B

3. 判断题

(1) 错误　(2) 错误　(3) 正确　(4) 正确　(5) 错误

第 2 章

1. 填空题

(1) 比较简单，复杂或有特殊要求

(2) Inetpub\wwwroot，http:// Localhost，http://服务器域名

(3) Machine.config

(4) 开发

2. 选择题

(1) B　(2) A　(3) D

3. 判断题

(1) 错误　(2) 正确　(3) 错误　(4) 正确

第 3 章

1. 填空题

(1) Hyper Text Markup Language(超文本标记语言)，World Wide Web

(2) 文本，浏览器

(3) 浏览器，浏览器

(4) 有序列表，无序列表

2. 选择题

(1) A　(2) B　(3) D　(4) D　(5) C,　B

3. 判断题

(1) 错误　(2) 错误

第 4 章

1. 填空题

(1) Cascading Style Sheet, 元素的外观
(2) stylesheet, text/css
(3) 像素(px)、字体宽度(em)，百分数(%)
(4) C
(5) B
(6) 普通流，相对定位和绝对定位

2. 选择题

B, D, A, C

3. 判断题

(1) 正确　(2) 错误　(3) 错误　(4) 错误　(5) 正确

第 5 章

1. 填空题

(1) 浏览器本身，服务器
(2) HTML 4，CSS，事件，脚本
(3) Document Object Model，DHTML
(4) 浏览器，服务器，基于脚本

2. 简答题

(2) 一般有四种标记：<img>、<embed>、<object>和<bgsound>。

第 6 章

1. 填空题

(1) 对表单的支持, 动画设计, Ajax
(2) DOM
(3) 所有 div 标记后代中使用 content 类选择器的元素
(4) 给所有 p 标记的元素增添 class="myclass"选择器
(5) 给所有 p 标记的元素删除 class="anotherclass"选择器

(6) textexp

2. 选择题

(1) C (2) B (3) C

3. 判断题

(1) 错误 (2) 正确 (3) 正确 (4) 正确 (5) 错误

第 7 章

1. 填空题

(1) System.Web.UI.Page
(2) 代码分离模式，单一模式
(3) 命名空间，类名，partial
(4) aspx.cs

2. 判断题

(1) 正确 (2) 错误

第 8 章

1. 填空题

(1) TextMode
(2) AutoPostBack
(3) Redirect
(4) nn.ToString();
(5) double.Parse(ss) ;

2. 选择题

(1) B (2) D

3. 判断题

(1) 正确 (2) 正确 (3) 正确

第 9 章

1. 填空题

(1) 视图状态，应用程序状态，会话状态，Cookie 状态
(2) (string)Session["greeting"]
(3) Lock()，UnLock
(4) useUri，AutoDetect
(5) Session.Timeout = 60;

(6) Session.Abandon();

2. 选择题

(1) A　　(2) C

3. 判断题

(1) 正确　　(2) 正确　　(3) 错误　　(4) 正确

第 10 章

1. 填空题

(1) ControlToValidate
(2) ValidationExpress

2. 选择题

(1) B　　(2) D　　(3) B　　(4) C

3. 判断题

(1) 错误　　(2) 正确　　(3) 正确

第 11 章

1. 填空题

(1) new DataSet ()，new DataSet ("表名")

(2) Connection 连接类，Command 命令类，DataAdapter 数据适配器类，DataReader 数据读取类

2. 选择题

(1) B　　(2) B

3. 判断题

(1) 错误　　(2) 正确　　(3) 正确　　(4) 错误　　(5) 正确
(6) 错误　　(7) 正确

第 12 章

1. 填空题

(1) CompositeDataBoundControl
(2) 10

2. 判断题

(1) 错误　　(2) 错误

第 13 章

1. 填空题

(1) 待定参数名，类型，实际参数
(2) 选择，父表的 GridView
(3) Request.QueryString()

2. 选择题

(1) B (2) D

3. 判断题

(1) 正确 (2) 错误 (3) 错误

第 14 章

1. 选择题

(1) C (2) D (3) B

2. 判断题

(1) 正确 (2) 错误 (3) 正确

第 15 章

1. 填空题

(1) LayoutTemplate, ItemTemplate
(2) 【高级】
(3) DataPager
(4) <%# Eval("age") %>

2. 选择题

(1) A, D, B, C
(2) C
(3) D, B

3. 判断题

(1) 正确 (2) 错误 (3) 正确 (4) 正确

第 16 章

1. 填空题

(1) 数据库

(2) T-SQL
(3) 参数名及类型，SQL 语句
(4) 缓存持续时间为 40 秒，与版本相关的属性为 none

2. 选择题

(1) D　　(2) B

3. 判断题

(1) 错误　　(2) 正确　　(3) 错误

第 17 章

1. 填空题

(1) App_Code
(2) 给数据表进行分页和排序，缓存数据，防止数据访问中的冲突

2. 选择题

(1) A　　(2) D　　(3) A

3. 判断题

(1) 正确　　(2) 错误

第 18 章

1. 填空题

(1) 匿名方法
(2) LINQ to Object, LINQ to XML, LINQ to SQL, LINQ to DataSet, LINQ to Entities

2. 选择题

(1) C　　(2) D

3. 判断题

(1) 错误　　(2) 正确　　(3) 正确

第 19 章

1. 填空题

(1) 服务器控件
(2) TextBox, Orang, DarkGreen
(3) 用户控件的命名空间(前缀)

2. 选择题

(1) D　　(2) D　　(3) A　　(4) D

3．判断题

(1) 正确　(2) 正确　(3) 正确　(4) 错误　(5) 正确

第 20 章

1．填空题

(1) 节点展开和折叠
(2) 深度为 3 层

2．选择题

(1) C　(2) A　(3) B

3．判断题

(1) 错误　(2) 错误　(3) 错误　(4) 正确

第 21 章

1．填空题

(1) 当前的登录状态
(2) 用户姓名
(3) 7 个字符以上，一个以上字母，一个以上非数字亦非字母的特殊符号
(4) PasswordRecovery
(5) ChangePassword

2．选择题

(1) C　(2) C　(3) B

3．判断题

(1) 正确　(2) 错误　(3) 错误

第 22 章

1．填空题

(1) 识别和管理用户信息，确定和保存用户关心的数据，允许用户自己定制网页界面
(2) enabled="true" />
(3) Name，string
(4) allowAnonymous="true"/

2．判断题

(1) 正确　(2) 错误　(3) 正确

第 23 章

1. 填空题

(1) Asynchronous JavaScript XML
(2) Ajax 引擎
(3) ActiveXObject
(4) XMLHttpRequest

2. 选择题

(1) C (2) D (3) B (4) D

3. 判断题

(1) 错误 (2) 正确

第 24 章

1. 选择题

(1) C (2) B (3) D (4) C

2. 判断题

(1) 错误 (2) 错误

第 25 章

1. 判断题

(1) 错误 (2) 错误 (3) 正确

第 26 章

1. 填空题

(1) 松耦合
(2) 通信的通用标准，服务与消费双方的协议
(3) Simple Object Access protocol
(4) 信息服务，计算
(5) 注册
(6) XML，服务的各项要求

2. 选择题

(1) A (2) D

3. 判断题

(1) 错误 (2) 错误 (3) 正确 (4) 错误 (5) 错误

第 27 章

1. 填空题

(1) Header 模板，Insert 模板
(2) ItemInserted
(3) HTML 的 Reset 按钮，服务器按钮
(4) ItemInserting

2. 选择题

(1) C (2) D (3) B

第 28 章

1. 填空题

(1) Global.asax.cs (2) CheckBox

2. 判断题

(1) 正确 (2) 正确 (3) 正确 (4) 错误

第 29 章

1. 填空题

(1)
A. 创建数据表对象 Cart
B. 如果购货车 ShoppingCart2 还没有建立
C. 从 Session 对象中取出数据表
D. 调用校验方法检查是否输入了客户标志
E. 取出 TextBox1 对象
F. 将取出的单价字段转换成浮点类型
G. 生成新的行对象
H. 将数据表存入 Session 对象中
(2)
A. 对 GridView 控件中的数据表的记录进行循环
B. 取出记录中的复选框控件
C. 如果复选框控件被选中
D. 求第八个字段的总和
E. 将第八个字段的总和在 TextBox 控件中显示出来

(3) <%# Eval("productID") %>

(4) <%# Eval("Color") %>, false

(5) ss.Split('.');

2. 选择题

(1) B　　(2) D　　(3) B　　(4) C

参 考 文 献

1. 单东林，张晓菲，魏然. 锋利的 jQuery. 北京：人民邮电出版社，2009
2. Dave Crane、EricPascarello (译者:李錕). Ajax 实战. 北京：人民邮电出版社，2006
3. Paolo Pialorsi. Programming LINQ(电子书)，http://www.pudn.com/downloads164/ebook/detail745514.html，2009
4. Stephen Walther. ASP.NET 3.5 Unleashed(电子书)，http://www.worldcat.org/title/aspnet-35-unleashed/oclc/213482738?referer=list_view，2008